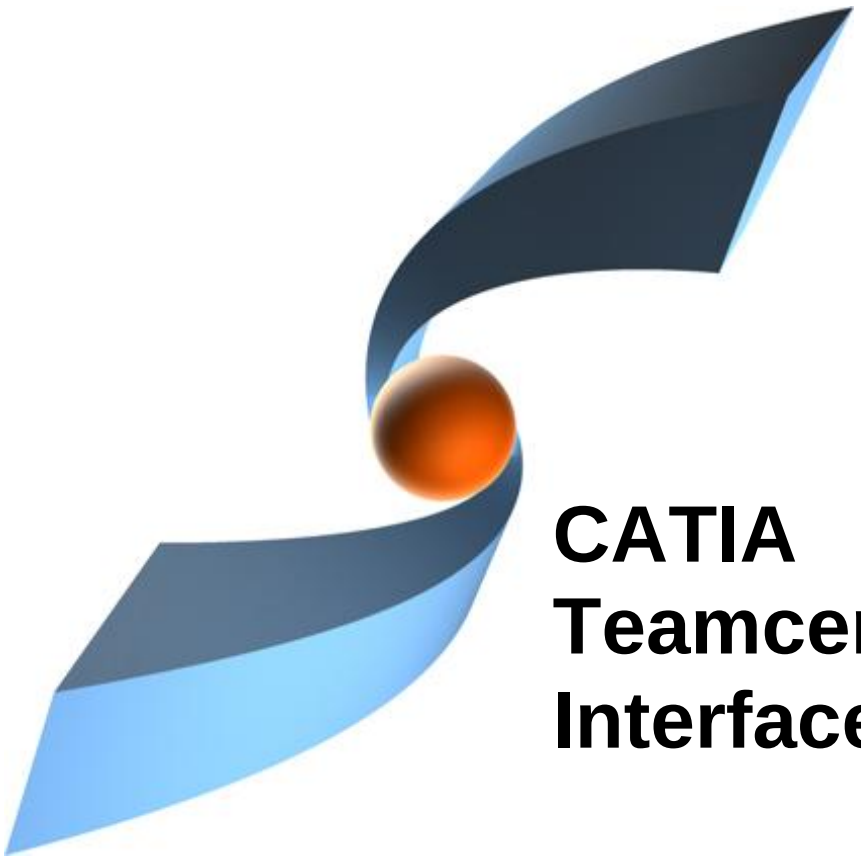

CATIA Teamcenter Interface

CMI Release 10.8

Customizing Guide

Copyright

© 1999, 2015 T-Systems International GmbH.
All rights reserved. Printed in Germany.

Contact

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

http://www.cmi-support.com

☎ +49 (0)40 5395 – 2020
🖷 +49 (0)391 5801 – 25688
mail : *cmi_support@t-systems.com*

Manual History

Version	Date	Version	Date
1.0	November 1996	8.9	March 2006
2.0	February 1997	9.0	October 2006
3.0	July 1997	9.1	March 2007
4.1	March 1998	9.2	October 2007
4.2	December 1998	9.3	March 2008
4.3	May 1999	9.4	October 2008
7.0	September 1999	9.5	March 2009
7.1	April 2000	9.6	October 2009
7.2	July 2000	9.7	April 2010
7.3	September 2000	9.8	October 2010
7.4	December 2000	9.9	April 2011
8.0	August 2001	10.0	October 2011
8.1	December 2001	10.1	April 2012
8.2	July 2002	10.2	October 2012
8.3	January 2003	10.3	April 2013
8.4	July 2003	10.4	October 2013
8.5	March 2004	10.5	April 2014
8.6	September 2004	10.6	October 2014
8.7	April 2005	10.7	April 2015
8.8	September 2005	10.8	October 2015

This edition obsoletes all previous editions.

Trademarks

CATIA is a registered trademark of Dassault Systèmes.

Teamcenter Enterprise is a registered trademark of Siemens PLM Corporation.

Names of other products mentioned in this manual are used only for identification purpose and may be trademarks of their companies.

Preface

About this Guide

This guide describes customizing and advanced configuration information for the CATIA Teamcenter Interface (CMI) with its two Teamcenter Enterprise servers generic Workbench and CATIA Workbench. Before using this guide, be sure you understand:

the UNIX-based operating system

the administration of the CATIA system

the administration of the Teamcenter Enterprise system

Related Documents

The following manuals contain information about installation, usage and customizing of CATIA Teamcenter Interface:

Manual Title	Version
CATIA Teamcenter Interface Installation & Administration Guide	10.8
CATIA Teamcenter Interface User Manual	10.8
CATIA Teamcenter Interface Customizing Guide	10.8

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

mail: *cmi_support@t-systems.com*

Table of Contents

CHAPTER 1 ..... **1**

OVERVIEW **1**

CHAPTER 2 ..... **3**

SYSTEM ARCHITECTURE **3**

CHAPTER 3 ..... **5**

ASSEMBLY STRUCTURE **5**

PREDEFINED OBJECT STRUCTURE 5
ENHANCED ASSEMBLY STRUCTURE RELATION OBJECTS 6
CLASS DESCRIPTION OF G2ASMPOS 7
CLASS DESCRIPTION OF G2ASMNPO 8
CLASS DESCRIPTION OF X2ASMPOQ 9

CHAPTER 4 ..... **11**

CUSTOMIZATION OPTIONS **11**

MESSAGE ACCESS RULES 11
Restrict permission to load a model into CATIA *11*
Restrict permission to modify a model from CATIA *11*
Restrict permission to modify Assembly positions from inside CATIA *11*
CLASS CONSTANTS 11
COMMON CUSTOMIZATION TASKS 12
Use of Structured Documents (eg. StDocmnt) *12*
Use a single Document per Part, for multiple CATParts *13*
Getting rid of the CATProduct Document *15*
Exclude objects or classes from the CMI Workbench *15*
Fill in custom attributes at a CATIA-Item in the Workbench *16*
Fill in custom attributes at a Document Representant in the Workbench *16*
Fill in custom attributes at a Model Representant in the Workbench *16*
Perform additional actions after a model was updated from CATIA *16*
Customize the name of parts in CATIA *17*
Customize for sending custom attributes to CATIA for Parts, Relations ,Models *17*
Show Teamcenter meta data in CATIA *17*
Customize CATProduct Worklocation *18*
Customize the CATDrawing filename *18*
Customize the CATPart filename *19*
Customize support CATProduct in CATIA V5 ..... *19*
Customize the CATProduct document (x0PrdDoc) *19*
Customize the CATPart document (GenDoc) *19*
Customize the Creation of the Part-Part relation during a Synchronize "Link Child" operation. *20*
Customize the Black Box functionality (deprecated) *20*
Customize the Component CATPart Data Model functionality *20*
Customize the Deletion of CATPart and CATProduct instances during Synchronize *21*
Define your own model types *21*
Filter Parts that are sent to Catia or VisMockup *21*
CAA CUSTOMIZATION 22
Configuration *23*

CHAPTER 5 ..... **25**

OPTIONAL CMI-FEATURES **25**

PLOTTING A MODEL 25
Conventions *25*
Sample for the definition of some naming conventions *26*

Creation of a CATIA-Model *27*
Definition of the relevant attributes *27*
LINX CUSTOMIZING METHODS 29
From LINX to CATIA *29*
STANDARD PROPERTIES IN CATIA V5 29
Customization: Sending standard attributes to CATIA V5 ..... *29*
Customization: Receiving user defined attributes from CATIA V5 ..... *29*
USER DEFINED PROPERTIES IN CATIA V5 30
Customization: Sending user defined attributes to CATIA V5 ..... *31*
Customization: Receiving user defined attributes from CATIA V5 ..... *31*
Customization: using CATIA V5 Properties during Part creation or model registration *32*
CONFIGURABLE BEHAVIORS IN CATIA V5 33
Descriptions of the behaviors: *34*
USAGE OF DATABASE NAME OF OBJECTS 35
Customization messages *35*
READ REFERENCE/SHEET INFORMATION FROM CATIA V5 DRAWINGS 37
Reference Documents for Drawings *37*
Sheet-Information for Drawings *37*
ENHANCED 4D-NAVIGATOR INTEGRATION 37
CATIA PROJECT ENVIRONMENT SUPPORT 37
SUPPORT OF TEAMCENTER "QUANTITY" 38
WORKING WITH CATIA V5 RELEASED CACHE 38
Configuration: *39*
CATPROCESS CUSTOMIZATION 40
customization methods in TeamCenter *40*
VIEWER SUPPORT 41
Prerequisites *41*
Features *41*
Configuration *42*
Customization *43*
Display User Data *44*
DESIGN TABLE SUPPORT 44
Features *44*
Configuration *45*
Customizable Methods *45*
MML SUPPORT 46
Features *46*
Installation *46*
Configuration *46*
Customizable Methods *47*
REPRESENTATION FORMATS IN CATIA V5 47
AUTOMATIC UPDATE OF CATDRAWING TITLE BLOCKS WITH TEAMCENTER DATA 48
TRANSFER OF WEIGHT PROPERTIES (INERTIA) FROM CATIA V5 TO TEAMCENTER 49
SET BOM-TYPE OF NEW CATIA-FILES BY TEAMCENTER-CUSTOMIZATION 50
CUSTOM EXPAND IN THE CMI CATIA WORKBENCH 51
Customization example *52*
SUPPLY ATTRIBUTES FROM CATIA TO TEAMCENTER AND BACK 54
Supply Attributes from Catia to Teamcenter *54*
Supply Attributes from Teamcenter to Catia *54*
SUPPLY FILE FROM CATIA TO TEAMCENTER 55
Supply File from Catia to Teamcenter *55*
POST -PROCESS CATPRODUCTS FOR SYNCHRONIZE / UPDATE 55
ENABLE PDM-CENTRIC SYNCHRONIZE 56
Validate actions before PDM- Centric Synchronize *56*
Often used slots in the NVSET's given in the messages above *56*
REFERENCE GEOMETRIES 57
VALIDATE CATIA VERSION 59
EXPORT TO FOLDER 60
Name customization for Export *60*
HANDLING OF MAPPING FILES 62
CATSCRIPT SUPPORT 64

PRODUCT BOUNDING BOXES 64
Configuration *65*
Customizing *65*

CHAPTER 6 67

WORKBENCH ARCHITECTURE 67

GENERIC WORKBENCH 67
CATIA WORKBENCH 67

CHAPTER 7 69

TEAMCENTER ENTERPRISE CONFIGURATION VARIABLES 69

CMI SERVER OPTIONS 69

CHAPTER 8 73

CATIA V4 DIRECTORY STRUCTURE 73

DIRECTORIES 73
FILES 74
MODIFY CATIA V4 ENVIRONMENT 75
USER DEPENDENT CONFIGURATIONS 76
USER DEPENDENT API IN CATIA 81

CHAPTER 9 83

CATIA V5 DIRECTORY STRUCTURE 83

DIRECTORIES 83
FILES 84
CUSTOMER DEPENDENT CONFIGURATIONS FOR CATIA V5 86
Environment settings *86*
Hide CMI Commands *93*

CHAPTER 10 95

DATA MODELS 95

DATA STRUCTURE CATIA-WORKBENCH 95
DATA STRUCTURE OF CMI-CLASSES 96
g0GenBin Class Hierarchy *97*
g0RepItm Class Hierarchy *98*

Table of Figures

FIGURE 1: CATIA WORKBENCH 1
FIGURE 2: SYSTEM ARCHITECTURE OF CMI 3
FIGURE 3: CMI OBJECT STRUCTURE 5
FIGURE 4: "MODEL HAS PLOT FILE" RELATION 5
FIGURE 5: AN EXAMPLE ASSEMBLY STRUCTURE 6
FIGURE 6: DATA MODEL EXTENSIONS FOR POSITIONING 7
FIGURE 7: ONE DOCUMENT FOR EACH CATPART 14
FIGURE 8: ONE DOCUMENT FOR ALL CATPARTS 14
FIGURE 9: ONE DOCUMENT FOR EACH CATPART 14
FIGURE 10: CATPART AND CATPRODUCT IN SAME DOCUMENT 15
FIGURE 11: SIMPLE DRAFT WITH DRAWING FRAME CONVENTIONS 25
FIGURE 12: USER DEFINED PROPERTIES 30
FIGURE 13: DATA MODEL EXTENSIONS FOR SUPPORT OF 'QUANTITY' 38
FIGURE 14: CATIA OPTIONS - CACHE MANAGEMENT 39
FIGURE 15: MODEL FILTER PREFERENCES DIALOG 42
FIGURE 16: CHANGE VIEWER PREFERENCE 42
FIGURE 17: VIEWER PREFERENCES DIALOG 42
FIGURE 18: MML SUPPORT BUTTONS 46
FIGURE 19: REPRESENTATION FORMATS IN CATIA V5 (CATIA WORKBENCH) 47
FIGURE 20: REPRESENTATION FORMATS IN CATIA V5 (CATIA V5) 48
FIGURE 21: CATDRAWING TITLE BLOCK 48
FIGURE 22: PROPERTIES WITH INERTIA 49
FIGURE 23: REFERENCE GEOMETRY 58
FIGURE 24: CLASS HIERARCHY OF PELMSTRC RELATION 58
FIGURE 25: EXPAND WITH REFERENCE 59
FIGURE 26: EXPORT TO FOLDER 60
FIGURE 27: CATEDM INSTALLATION PATH STRUCTURE 73
FIGURE 28: INITIALIZATION FILES WITH THEIR ORDER. 76
FIGURE 29: DIRECTORY STRUCTURE OF THE CMICATV5 MODULE 83
FIGURE 30: DIRECTORY STRUCTURE OF THE CMICATV5 INSTALLATION DIRECTORY 84
FIGURE 31: EXAMPLE OF DIRECTORY STRUCTURE OF THE CMICATV5 INSTALLATION SUBDIRECTORY MSGCATALOG 86
FIGURE 32: SYNCHRONIZE DIALOG WITH CATIA NODE NAMES 94
FIGURE 33: CATIA OPTIONS - NODES CUSTOMIZATION 94
FIGURE 34: DATA STRUCTURE OF CATIA-WORKBENCH 95
FIGURE 35: DATA STRUCTURE OF CMI-CLASSES 96
FIGURE 36: CLASS HIERARCHY UNDER G0GENBIN 97
FIGURE 37: CLASS HIERARCHY UNDER G0REPITM 98

CHAPTER 1

Overview

Individual parts and multiple-layer assemblies can now be processed directly in CATIA thanks to the integration of CATIA into Teamcenter Enterprise; for this the CATIA data is under the control of Teamcenter Enterprise. The CATIA-Workbench in Teamcenter Enterprise serves as the communication medium between the PDM-System Teamcenter Enterprise and the CAD-System CATIA. This is a special structure browser which displays the current storage contents of the CATIA session graphically and provides manipulation facilities.

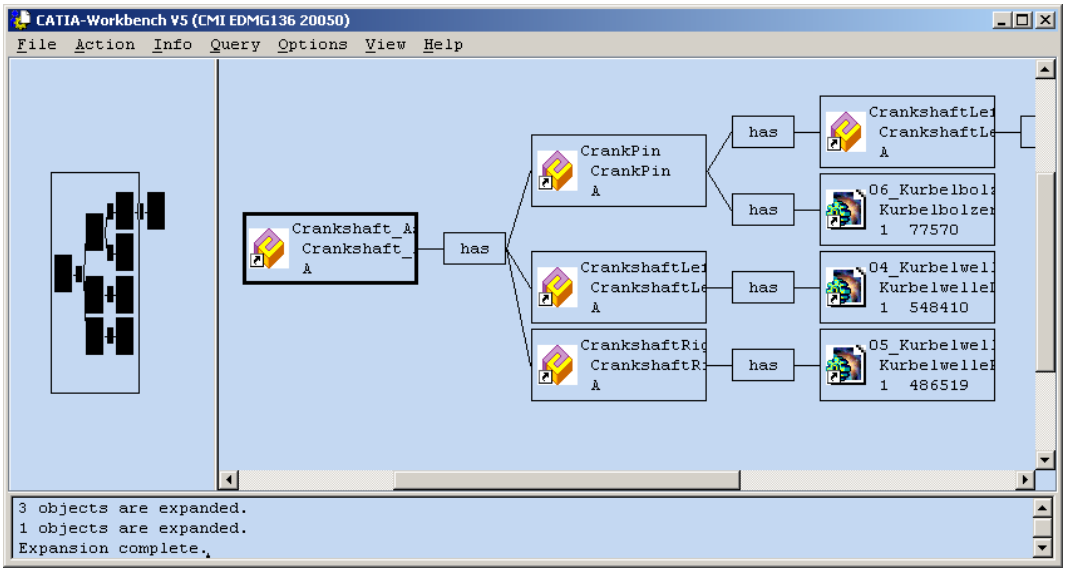

Figure 1: CATIA Workbench

The entire data interchange between Teamcenter Enterprise and CATIA is effected via the Workbench.

CHAPTER 2

System Architecture

The components of CMI are:

GMI	Generic Workbench defines some base classes (see „Data Structure of Catia-Workbench“ on page 956)
CMI	CATIA Workbench defines CATIA specific classes and allows the interaction between the user and CATIA (see „Data Structure of “ on page 967).
Listener	Listener manages the communication between CATIA and Teamcenter Enterprise(Collaboration Foundation, Metaphase).
CATEDM	The CATIA V4 GII module provided by T-Systems Enterprise Services GmbH allows CATIA V4 to manage multiple-level assemblies.
CMICATV5	The CATIA V5 module provided by T-Systems Enterprise Services GmbH allows CATIA V5 to be integrated into Teamcenter Enterprise.
Exchange Map	A dedicated user directory on the client workstation. The CATIA extension expects the model files to be only within this directory.

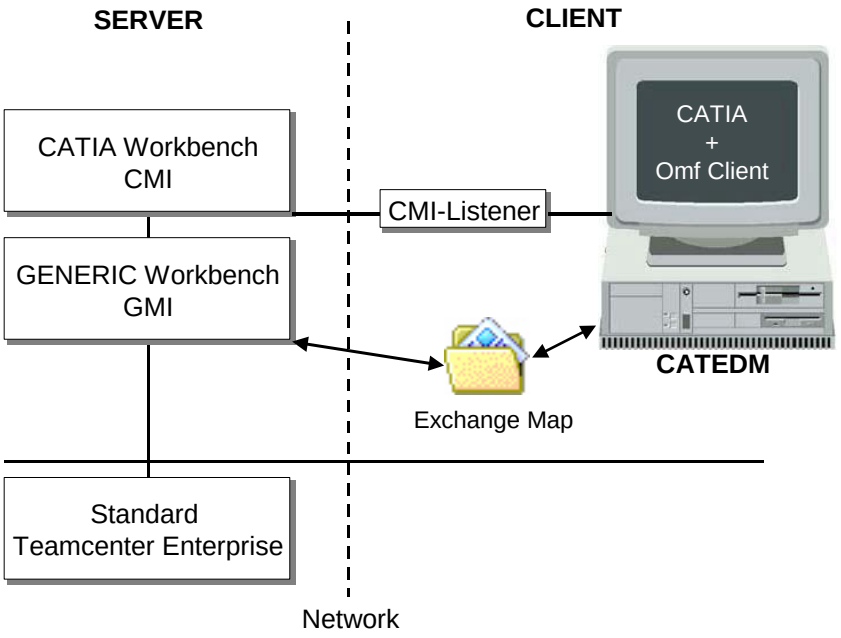

Figure 2: System architecture of CMI

The interaction starts either on the CATIA side or in Teamcenter Enterprise. The listener allows the communication between the Teamcenter Enterprise client and CATIA session. They communicate through RPC and ToolTalk protocol.

CHAPTER 3

Assembly Structure

Predefined Object Structure

The *CATIA Metaphase Interface* contains the following pre-defined object structure:

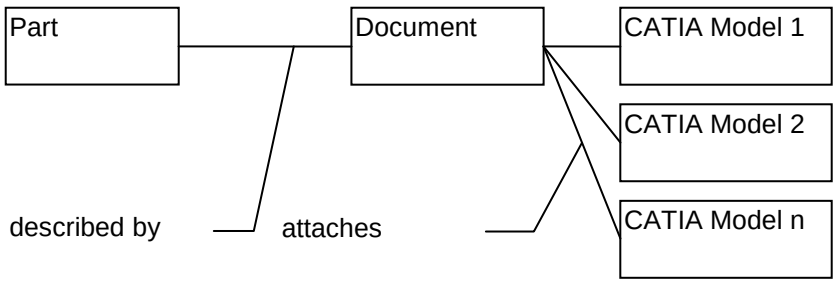

Figure 3: CMI object structure

For each part only one relation „**described by**“ with a document element has to exist. The relationship can be created by dragging any document object and dropping it to the part object. Please refer to the Teamcenter Enterprise user documentation for managing relationships between Teamcenter Enterprise objects.

The documents can contain at least one relation „**attaches**“ with CATIA model elements. You can attach more than one CATIA model to a certain document.

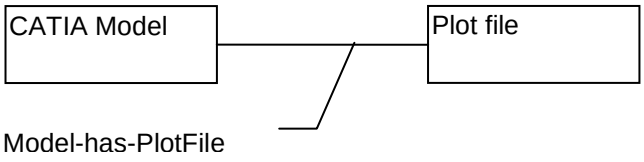

Figure 4: "Model has plot file" relation

Each certain CATIA model object can contain one relation „**Model has plot file**“ with any plot file, which can be created from CATIA.

An example structure is shown in the following figure:

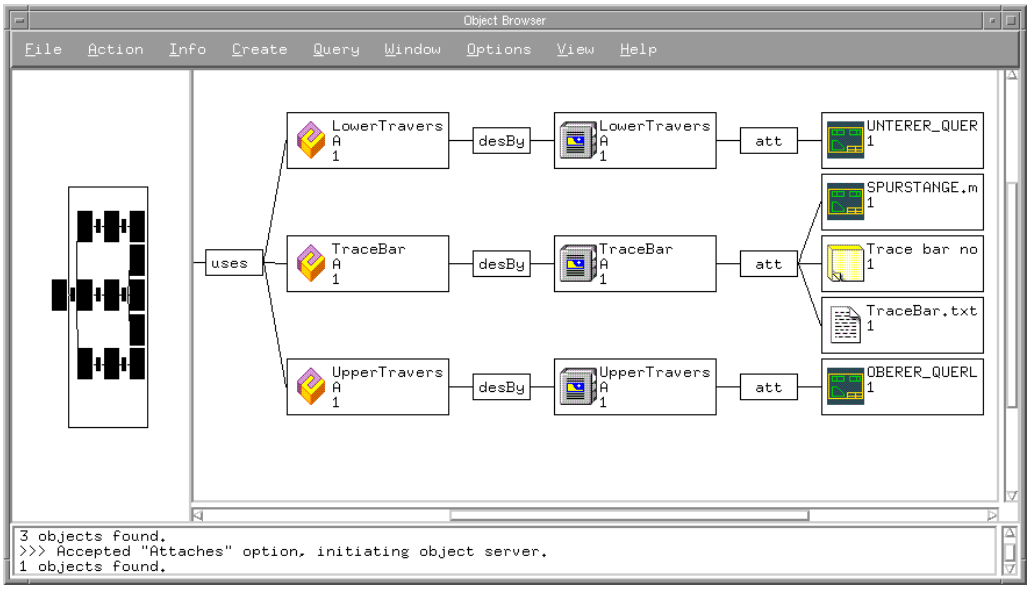

Figure 5: An example assembly structure

If you drag a part object and drop it into the CATIA Workbench, only the CATIA relevant data will be extracted and shown in the CATIA Workbench window
(see Figure 1: CATIA Workbench on page 1).

The following chapters describe the customizing possibilities to influence the behavior of the CATIA Workbench.

☞ **It is recommended to consult the custom/README file in order to ensure the actuality of source code and customizing tasks.**

Enhanced Assembly Structure Relation Objects

In order to manage assembly structures with and without position information, CMI extends the standard Teamcenter Enterprise data model with three new classes. "**`g2AsmRel`**" is used in order to define the general behavior of the CMI assembly structure relation objects which can be found directly below the standard Teamcenter Enterprise class "AssmStrc". "**`g2AsmNPo`**" defines the assembly structure relation object without position information and "**`g2AsmPos`**" for assembly structures with position information. Both classes are derived from "g2AsmRel".

1. g2AsmRel - defines the general behavior of CMI assembly structure relation objects
2. g2AsmNPo - for assembly structures without position information
3. g2AsmPos - for assembly structures with position information

The Teamcenter Enterprise data model with CMI looks like this :

1 a Structur "Structure"

2 p AssmStrc "Assembly Structure"

3 a **g2AsmRel** "CMI Assembly Structure Relation"

4 p **g2AsmNPo** "CMI Assembly Structure without position"

4 p **g2AsmPos** "CMI Assembly Structure with position"

If your specific assembly structure relation is derived from g2AsmPos (assembly structure relation with position information) some additional methods are called for the validation of this information. Due to the fact that validation routines always need some time to perform their actions, it is necessary to decide if the new relation object needs position information or not. If yes, you need to derive this class from g2AsmPos, if not you should use g2AsmNPo.

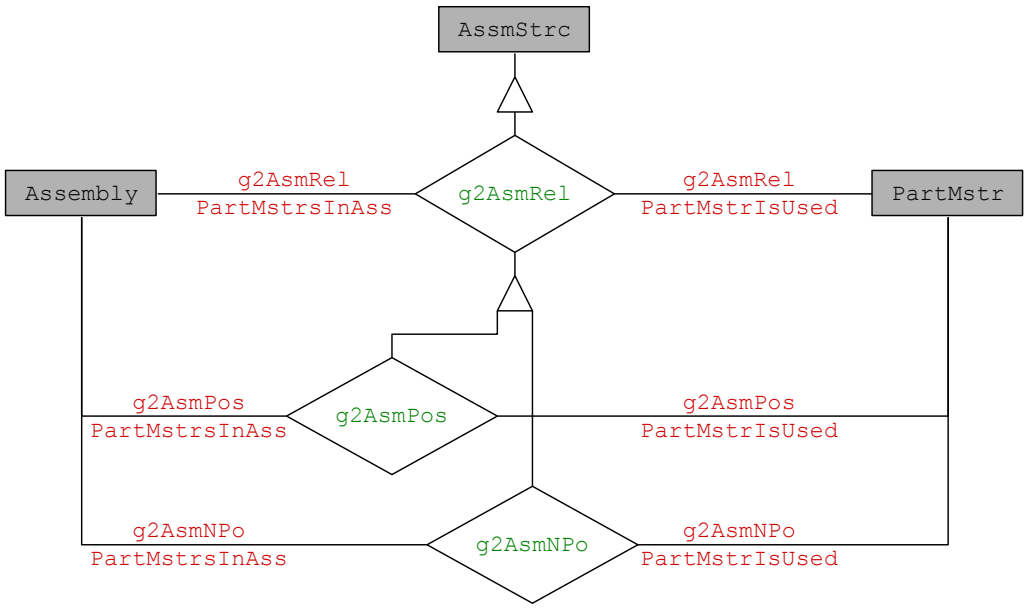

Figure 6: Data model extensions for positioning

Class Description of g2AsmPos

```
                          Class : g2AsmPos
                         Parent : g2AsmRel
                         Prompt : "g2AsmPos"
                  Left Side Class : Assembly

                 Right Side Class : PartMstr

          Forward Relationship Name : g2AsmPosPartMstrsInAss
                                      (droppable)

         Backward Relationship Name : g2AsmPosPartMstrIsUsed
                                      (droppable)
```

Attributes	Opts	Type	Prompt
TMatrix11		string(30)	"TMatrix11"
TMatrix12		string(30)	"TMatrix12"
TMatrix13		string(30)	"TMatrix13"
TMatrix14		string(30)	"TMatrix14"
TMatrix21		string(30)	"TMatrix21"
TMatrix22		string(30)	"TMatrix22"
TMatrix23		string(30)	"TMatrix23"
TMatrix24		string(30)	"TMatrix24"
TMatrix31		string(30)	"TMatrix31"
TMatrix32		string(30)	"TMatrix32"
TMatrix33		string(30)	"TMatrix33"

```
TMatrix34                    string(30)    "TMatrix34"
TMatrix41                    string(30)    "TMatrix41"
TMatrix42                    string(30)    "TMatrix42"
TMatrix43                    string(30)    "TMatrix43"
TMatrix44                    string(30)    "TMatrix44"

Opts: i = inherited, d = dynamic, c = cached, r = required


            Constants          Inherit    Value
------------------------------ ------- ------------

GetInfoDialogC                   no    x1AtcGet
g4TrafoTypeC                     no    4x4


         Messages      Opts        Type
---------------------- ---- -------------------
g3CreateOBID            i   ObjectMessage
g3CreateRelationObj     i   ClassMessage
g3CreateWidgetId        i   ObjectMessage
g3DeleteObject          i   ObjectMessage
g3DeleteRelationObject  i   ObjectMessage
g3GetTrafo              i   ObjectMessage
g3UpdateTrafo           i   ObjectMessage
g3ValForUpdTrafo        i   ObjectMessage
```

Class Description of g2AsmNPo

Class : g2AsmNPo

Parent : g2AsmRel

Prompt : "g2AsmNPo"

Left Side Class : Assembly

Right Side Class : PartMstr

Forward Relationship Name : g2AsmNPoPartMstrsInAss (droppable)

Backward Relationship Name : g2AsmNPoPartMstrIsUsed (droppable)

```
            Constants          Inherit    Value
------------------------------ ------- ------------
g4TrafoTypeC                     yes   NO


         Messages      Opts        Type
---------------------- ---- -------------------
g3CreateOBID            i   ObjectMessage
g3CreateRelationObj     i   ClassMessage
g3CreateWidgetId        i   ObjectMessage
g3DeleteObject          i   ObjectMessage
g3DeleteRelationObject  i   ObjectMessage
g3GetTrafo              i   ObjectMessage
g3UpdateTrafo           i   ObjectMessage
g3ValForUpdTrafo        i   ObjectMessage
```

Class Description of x2AsmPoQ

Class : x2AsmPoQ

Parent : g2AsmPos

Prompt : "x2AsmPoQ"

Left Side Class : Assembly

Right Side Class : x0AssmMr

Forward Relationship Name : x2AsmPoQPartMstrsInAss (droppable)

Backward Relationship Name : x2AsmPoQPartMstrIsUsed (droppable)

All Attributes	Opts	Type	Prompt
x0TrafoList		Table	"Transformation List"

Opts: i = inherited, d = dynamic, c = cached, r = required

, All Constants	Inherit	Value
GetInfoDialogC	no	DAtcGetI
PsmAssocStRevRevClassC	no	g2ARvPos
UpdateDialogC	no	x1APQUpd

All Messages	Opts	Type
DoCreateRelPost	o	ObjectMessage
DoUpdatePre	o	ObjectMessage
g3CreateOBID	i	ObjectMessage
g3CreateRelationObj	i	ClassMessage
g3CreateWidgetId	i	ObjectMessage
g3DeleteObject	i	ObjectMessage
g3DeleteRelationObject	i	ObjectMessage
g3GetOccName	i	ObjectMessage
g3GetTrafo	i	ObjectMessage
g3UpdateTrafo	i	ObjectMessage
g3ValForUpdTrafo	i	ObjectMessage
x3AddTrafo		ObjectMessage
x3CheckForMultiOcc	i	ObjectMessage
x3DelAssmStrcRel	o	ObjectMessage
x3DeleteTrafo		ObjectMessage
x3GetTrafos	o	ObjectMessage
x3GetTrafosOfTable		ObjectMessage
x3UpdateTrafo	o	ObjectMessage

CHAPTER 4

Customization Options

Message Access Rules

With message access rules you can control read and write access to CMI data based on properties of the user or the data itself.

Restrict permission to load a model into CATIA

A user needs access to the **"View"** message for *x0CTFile* objects in order to load model files under Teamcenter Enterprise control into CATIA. The default rules allow any user to view any model.

Restrict permission to modify a model from CATIA

A user needs access to the **"Edit"** message for *x0CTFile* objects in order to modify model files under Teamcenter Enterprise control from CATIA. The default rules don't allow a user to update a model that doesn't belong to him

Restrict permission to modify Assembly positions from inside CATIA

A user needs access to the **"g3UpdateTrafo"** message for *Part* objects in order to modify Assembly positions. The default rules don't allow a user to update a position of a part if the parent assembly doesn't belong to him.

Class Constants

There are some class constants that you can override to meet the specific needs of your customization.

```
define value set g0RelsOfPart ["PartDoc",
"DocumentsDescribingPart"];

Part.met:Part.g4RelsC = "g0RelsOfPart";
```

This determines which Part-Document relationships are searched for documents. If your Parts are described by a large variety of documents but few attach CATIA models, you may want to use a dedicated relation class for "CATIA documents". By overriding this class constant you can make this relation class known to CMI, so it will only search for documents with this relation to the Part.

```
define value set
g0RelsOfDoc["Attach","DataItemsAttachedToBusItem"];

GenDoc.g4RelsC= "g0RelsOfDoc";
```

This determines which Document-DataItem relations/relationships will be searched for CATIA models. If your documents attach a large number of files other than CATIA models, you may want to use a dedicated relation class for the CATIA models. By overriding this class constant you can make this relation class known to CMI, so it will only search for files with this relation to the document.

```
define class constant x4CreateCatPartC;

x0WkBnch.met: x0WkBnch.x4CreateCatPartC = x0CatPrt;
```

This determines which class is used to create a CATPart in Teamcenter Enterprise from CATIA V5. If you want to create a CATPart in Teamcenter Enterprise with another class than class **x0CatPrt**, you should override this class constant by the name of the other class. This new class must be a child class of **x0CatPrt**.

```
define class constant x4CreateCatDrawC;

x0WkBnch.met: x0WkBnch.x4CreateCatDrawC = x0CatDrw;
```

This determines which class is used to create a CATDrawing in Teamcenter Enterprise from CATIA V5. If you want to create a CATDrawing in Teamcenter Enterprise with another class than class **x0CatDrw**, you should override this class constant by the name of the other class. This new class must be a child class of **x0CatDrw**.

Common Customization Tasks

Use of Structured Documents (eg. StDocmnt)

Since CMI 8.9 you can use structured documents classes without writing any method code. It is sufficient to set the corresponding class constants in the data model. You can use any other document class in the same way.

The following example shows use of `StDocmnt / PrtSDocR` (Part to Document Master relation). Also, in this example the same class is used for the Product Document that is used for “regular” documents.

Example:

```
O:x0WkBnch.x4CreatePrdDocC    = StDocmnt;

O:x0WkBnch.x4PartToPrdDocC    = PrtSDocR;

O:x0WkBnch.g4PrdDocClassC    = StDocmnt;

O:x0WkBnch.x4CmpDocClassC        = StDocmnt;

O:x0WkBnch.x4CmpPartDocRelClassC = PrtSDocR;

O:x0WkBnch.x4CmpDocPartRelShipC  = AddedInfoForParts;

O:x0WkBnch.g4BlackBoxDocClassC     = StDocmnt;

O:g0GenWB.g4DocClassC       = StDocmnt;

O:g0GenWB.g4PrdDocClassC    = StDocmnt;

O:g0GenWB.g4PartPrdDocRelC  = PrtSDocR;

O:g0GenWB.g4PartPrdDocRSC   = AddedInfoDocuments;

O:g0GenWB.g4BlackBoxDocClassC     = StDocmnt;

O:g0GenWB.g4PartBlackBoxDocRelC   = PrtSDocR;
```

```
O:g0GenWB.g4PartBlackBoxDocRSC    = AddedInfoDocuments;


//Base Class of documents relevant for Catia V4/V5


define value set StrcRelsOfPart ["PrtSDocR",
"AddedInfoDocuments"];
O:Part.g4RelsC = "StrcRelsOfPart";


//Teamcenter "Prepare" Dialog for Catia Files should offer
//Structured Document for Prepare


O:g0GenBin.StartClassToPrepareC = "StGenDoc";
O:g0GenBin.DefaultClassToPrepareC = "StDocmnt";
```

If you want to use the PartSDoc-relation(Part to Structured Document), the same example must be changed to:

```
O:x0WkBnch.x4CreatePrdDocC    = StDocmnt;
O:x0WkBnch.x4PartToPrdDocC    = PartSDoc;
O:x0WkBnch.g4PrdDocClassC    = StDocmnt;
O:x0WkBnch.x4CmpDocClassC        = StDocmnt;
O:x0WkBnch.x4CmpPartDocRelClassC = PartSDoc;
O:x0WkBnch.x4CmpDocPartRelShipC  = PartsRepresentedbyDocument;
O:x0WkBnch.g4BlackBoxDocClassC     = StDocmnt;
O:g0GenWB.g4DocClassC       = StDocmnt;
O:g0GenWB.g4PrdDocClassC    = StDocmnt;
O:g0GenWB.g4PartPrdDocRelC  = PartSDoc;
O:g0GenWB.g4PartPrdDocRSC   = DocumentRepresentingPart;
O:g0GenWB.g4BlackBoxDocClassC     = StDocmnt;
O:g0GenWB.g4PartBlackBoxDocRelC   = PartSDoc;
O:g0GenWB.g4PartBlackBoxDocRSC    = DocumentRepresentingPart;
define value set StrcRelsOfPart ["PartSDoc",
"DocumentRepresentingPart"];
O:Part.g4RelsC = "StrcRelsOfPart";
O:g0GenBin.StartClassToPrepareC = "StGenDoc";
O:g0GenBin.DefaultClassToPrepareC = "StDocmnt";
```

Use a single Document per Part, for multiple CATParts

In the OOTB Synchronize function in CATIA V5, where CMI creates new Parts and Documents, the user can create a fresh Document for each CATPart, or he may use an existing document at the Part for all CATPart files that are linked to the Part. This is effected by using the TC *Prepare* functionality, where either an existing or a new document name can be given by the user.

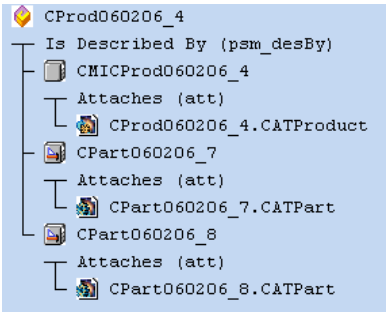

Figure 7: One Document for each CATPart

In an organization where there is always only one Document describing a Part, the following Teamcenter configuration variable can greatly enhance usability and consistency:

```
set CMI_SINGLE_PART_DOCUMENT "ON";
```

If this variable is set, only one document is created for each Part during Synchronize; all CATParts will be attached to this document. Also, if a Part already has a describing Document of the right class, it will be used.

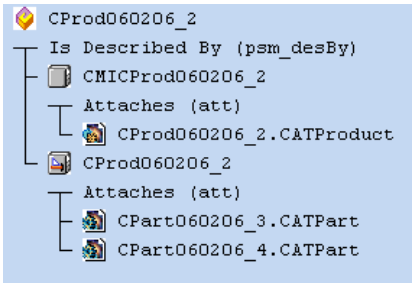

Figure 8: One Document for all CATParts

The complementary data model is obtained with the following setting

```
set CMI_SINGLE_PART_DOCUMENT "SINGLE_FILE";
```

This will create a new document for each CATPart, which will be created automatically and named after the CATPart.

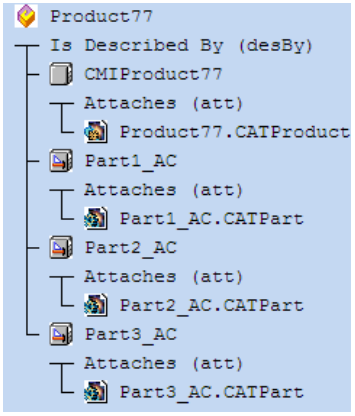

Figure 9: One Document for each CATPart

Getting rid of the CATProduct Document

In order to improve consistency CMI uses a special class and relation for the Document that attaches a CATProduct. In an OOTB CMI installation you will typically see two Documents: one that attaches the CATProduct and one that attaches the CATParts. This is because the CATProduct should not be exposed to direct user actions.

Using the TC configuration
set CMI_SINGLE_PART_DOCUMENT “ON”; (see above)

You can get rid of this special document. Just set the class constants pertaining to the Product Document to point to your regular Document class.

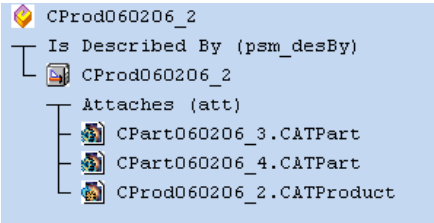

Figure 10: CATPart and CATProduct in same Document

The following example shows how to set the Product Document to the same document class used to attach CATParts (***DesDoc***)

Example:

```
O:x0WkBnch.x4CreatePrdDocC    = DesDoc;
O:x0WkBnch.x4PartToPrdDocC    = PartDoc;
O:x0WkBnch.g4PrdDocClassC    = GenDoc;
O:x0WkBnch.x4CmpDocClassC        = DesDoc;
O:x0WkBnch.x4CmpPartDocRelClassC = PartDoc;
O:x0WkBnch.x4CmpDocPartRelShipC  = DocumentsDescribingPart;
O:g0GenWB.g4PrdDocClassC    = GenDoc;
O:g0GenWB.g4PartPrdDocRelC  = PartDoc;
O:g0GenWB.g4PartPrdDocRSC   = DocumentsDescribingPart;
//make sure that the same Document class is used by the
//”Prepare” function
O:g0GenBin.StartClassToPrepareC = DesDoc;
O:g0GenBin.DefaultClassToPrepareC = DesDoc;
```

in config.cfg:

```
set CMI_SINGLE_PART_DOCUMENT “ON”;
```

Exclude objects or classes from the CMI Workbench

CMI calls the message **`g0PdmItm:g3CheckIfItemInScope`** to check whether it must handle a document or model object or ignore it. You can exclude an entire class or individual objects based on their attributes.

For Documents you can also override GenDoc:g3CheckIfDocInScope if you want to filter documents based on attributes of the Part they describe.

See also

g0PdmItm:g3CheckIfItemInScope

DataItem:g3CheckIfItemInScope

g0GenMod:g3CheckIfItemInScope

GenDoc:g3CheckIfDocInScope

in g3Custom.mth

Fill in custom attributes at a CATIA-Item in the Workbench

By default, all attributes of the Part are copied to the CATIA Item. So to add attributes from your Part class you only need to attach those attributes to **x0CTItem**.

To do additional work after a CATIA-Item is created, override **g0GenItm:g3CreateGIPost**.

To set attributes based on the **AsmStrc** relation between the part and its parent Assembly, override **g0GenItm:g3SetAttrsFromRelation**. Note that the passed relation may be NULL, if the CATIA Item is a root level assembly.

See also

g0GenItm:g3SetGIAttrs

g0GenItm:g3CreateGIPost

g0GenItm:g3SetAttrsFromRelation

in g3Custom.mth

Fill in custom attributes at a Document Representant in the Workbench

To fill in additional attributes based on the original Document after a Document Representant was created, override **g0DocRep:g3SetSpecificAttrs**

See

g0DocRep:g3SetSpecificAttrs

in g3Custom.mth

Fill in custom attributes at a Model Representant in the Workbench

To fill in additional attributes based on the original CATIA model after a Model Representant was created, override **x0ModRep:g3SetSpecificAttrs**.

Note that the default implementation is not empty, so you must call the parent method. This requires to derive your own class from **x0ModRep**.

See

x0ModRep:g3SetSpecificAttrs

in x3Custom.mth

Perform additional actions after a model was updated from CATIA

To perform additional actions after a CATIA model was modified from inside CATIA, override **x0CTFile:x3SaveModelPost**

See

x0CTFile:x3SaveModelPost

in x3Custom.mth

Customize the name of parts in CATIA

To customize the name of parts displayed in CATIA override **x0CTItem:x3GetDescriptionInCAD**. By default the part number is shown.

See

x0CTItem:x3GetDescriptionInCAD

in x3Custom.mth

Customize for sending custom attributes to CATIA for Parts, Relations ,Models

Override:

x0CTFile:x3GetCustomDataForCAD

Part: x3GetCustomDataForCAD

g2AsmPos: x3GetCustomDataForCAD

x0WkBnch:x3SendCustomAttrPref

in x3Custom.mth

Following you will find a detailed description which method has to be overwritten in which case:

Activate LINX interface

→ **x0WkBnch:x3SendCustomAttrPref in x3Custom.mth**

Send custom attributes from Model to CATIA

→ **x0CTFile:x3GetCustomDataForCAD in x3Custom.mth**

Send custom attributes from Part to CATIA

→ **Part:x3GetCustomDataForCAD in x3Custom.mth**

Send custom attributes from Relation between Assemblies to CATIA

→ **g2AsmPos:x3GetCustomDataForCAD in x3Custom.mth**

Show Teamcenter meta data in CATIA

The *More* Button in the CMI Info command allows to retrieve realtime information about the selected CATIA V5 item from Teamcenter. By default the *Get Item Info* dialogs define the information shown.

In CATIA V4 this information is displayed by the MODEL INFO panel.

You can customize the information that is displayed by overriding the following methods:

```
message g0GenBin:x3GetItemInfoForCAD (
   input : ObjectPtr    this           ::
   output: SetOfStrings *LabelSet      ::
   output: SetOfStrings *ValueSet      ::
```

```
   output: integer      *mfail) code
```

To retrieve information from the data item

```
message Part:x3GetPartInfoForCAD (
   input : ObjectPtr      this        ::
   input : NULL ObjectPtr PartRel     ::
   output: SetOfStrings   *LabelSet   ::
   output: SetOfStrings   *ValueSet   ::
   output: integer        *mfail) code
```

To retrieve information from the TC Part (V5 only)

```
message Relation:x3GetRelInfoForCAD(
   input : ObjectPtr    this           ::
   output: SetOfStrings *LabelSet      ::
   output: SetOfStrings *ValueSet      ::
   output: integer      *mfail) code
```

To retrieve information from the Assembly Structure Relation (V5 only)

Original implementation available in x3Custom.mth

Customize CATProduct Worklocation

Return the **x1PrdDataWorkloc** Attribute from the Workbench Object. The returned string must be freed with **nlsStrFree()**.

```
x0WkBnch:g3CatPrdV5Workloc (
   input : string   classname    ::
   output: string   *sCatPrdV5WL ::
   output: integer  *mfail)
```

Original implementation available in x3Custom.mth

Customize the CATDrawing filename

Customize the default filename of a new CATDrawing

```
x0CatDrw:x3CreateModelName (
   input  :      ObjectPtr this         ::
   input  :      string    pHost        ::
   input  :      string    pUserName    ::
   input  :      string    pDirectory   ::
   input  : NULL ObjectPtr DocObj       ::
```

```
   output :      string    *filename    ::
   output :      integer   *mfail)
```

Original implementation available in x3Custom.mth

Customize the CATPart filename

Customize the default filename of a new CATPart

```
x0CatPrt:x3CreateModelName (
   input  :      ObjectPtr this         ::
   input  :      string    pHost        ::
   input  :      string    pUserName    ::
   input  :      string    pDirectory   ::
   input  : NULL ObjectPtr DocObj       ::
   output :      string    *filename    ::
   output :      integer   *mfail)
```

in x3Custom.mth

Customize support CATProduct in CATIA V5

Supress CATProducts to CATIA V5

```
x0WkBnch:g3CatPrdV5Support (
   input : string   classname     ::
   output: boolean  *bCatPrdV5Sup   ::
   output: integer  *mfail)
```

Original implementation available in x3Custom.mth

Customize the CATProduct document (x0PrdDoc)

Customize the creation of CATProduct document

```
x0PrdDoc:x3CreatePrdDoc(
   input  : string       className         ::
   input  : string       partNumber        ::
   input  : ObjectPtr    partObj           ::
   output : ObjectPtr    *newPrdDocObj     ::
   output : integer      *mfail)
```

Original implementation available in x3Custom.mth

Customize the CATPart document (GenDoc)

Customize the creation of Model or CATPart document

```
Part:x3LinkWithDocument (
  input  : ObjectPtr  thisObject    ::
  input  : ObjectPtr  docObject     ::
  output : integer *mfail)
```

```
x0CTFile:x3CreateDocForModel (
  input      : ObjectPtr  thisObject     ::
  input NULL : ObjectPtr  partObject     ::
  output     : ObjectPtr  *newDocObject  ::
  output     : integer *mfail)
```

Original implementation available in x3Custom.mth

If partObject is NULL, then the document is being created at file-registration time (during a "Create" operation in Synchronize) and the parent Part is unknown. If you would prefer to rather create the document later (or to be able to select a document already attached to a Part) at the time when the File is to be attached to the Part, then you should change this method to block the creation of the document when the partObject is NULL.

Customize the Creation of the Part-Part relation during a Synchronize "Link Child" operation.

Customize the following message to your needs:

```
class message AssmStrc:x3CreateUsesRel(
   input : string       className         ::
   input : ObjectPtr    leftPrtObj        ::
   input : ObjectPtr    rightPrtObj       ::
   input : NvSet        usesInfos         ::
   output: ObjectPtr    *newRelObj        ::
   output: string       *trafoIndex       ::
      output: integer      *mfail)
```

in x3Custom.mth

Customize the Black Box functionality (deprecated)

The CATIA V5 Black Box creation/add/remove functionality uses the following API's:

`x0PrdDoc:x3AttachBlackBoxFiles()`	to attach the black box files to the document
`Part:x3CreBlackBoxFileObj ()`	to create a black box part file object
`Part:x3RemoveBlackBoxFiles ()`	to delete black box files
`Part:x3AddBlackBoxFiles ()`	to copy black box files to the user's work location and register them

in x3Custom.mth.

Customize the Component CATPart Data Model functionality

The CATIA V5 Component CATPart Data Model functionality uses the following API's:

`x0CatPrt:x3CreateCmpDataPost ()`	Operations after a x0CatPrt has been created in Teamcenter Enterprise (new Data Model)
`ProdBI:x3CreateCmpDocPost ()`	Operations after a Document has been created in Teamcenter Enterprise (new Data Model)

x0CatPrt:x3SaveComponentPost ()	Operations after a Part has been updated in Teamcenter Enterprise (new Data Model)

in x3Custom.mth.

Customize the Deletion of CATPart and CATProduct instances during Synchronize

This is a part of CATIA V5 Synchronize Command functionality and uses the following API's:

x0CTFile:x3ProcessCATIADeletion()	Called during the Interactive "Update and Create" procedure for each CATPart or V4 Model deleted from the product structure by the user in CATIA V5. In OOTB CMI, this is not implemented. Override to remove file instances during Synchronize.
Part:x3ProcessCATIADeletion ()	Called during the Interactive "Update and Create" procedure for each CATProduct removed from the product structure in CATIA V5.

in x3Custom.mth.

Define your own model types

If you'd like to have your own values for filtering models in the Catia-Workbench you can change the attribute **g0ModelType**. This attribute is also contained in the Query- and Create dialogs for Catia-Files. Then you define your own value set for **g0ModelType**. After that you have to set the class constant **g4ModelTypesC** to the name of your value set at the class **g0GenMod**. It is also necessary to attach this value set to the attribute **g0ModelType** with a condition.

Example:

```
o:g0GenMod.g4ModelTypesC="[your_value_set]";
```

It is necessary to have a condition for changing the value set at the attribute g0ModelType.

```
define condition ALWAYS_TRUE(obj) := "1=1";

attach value set [your_value_set] to g0ModelType if (ALWAYS_TRUE);
```

Filter Parts that are sent to Catia or VisMockup

You can filter parts, which should not be sent to Catia or to the viewer. If a part is filtered, the part itself and all its subtrees are not sent to Catia / to the viewer. The following method can be overwritten to filter parts:

```
class message x0WkBnch:x3FilterPartsToSend (
   input:      string          classname     ::
   input:      string          sTargetSystem ::
   update:     SetOfObjects    *parts        ::
   output:     integer         *mfail)
```

The parameter **sTargetSystem** can contain the values "**V4**", "**V5**" or "**Viewer**". This is the system where the parts are currently sent to. If objects are removed from the SetOfObjects parameter **parts**, they and their subtrees are not sent.
This method does not impact the display in the CMI Workbench.

CAA Customization

An API is provided for performing custom steps before and after the Synchronize function in CATIA, as callback functions in CAA. The callback is implemented by loading a customer compiled shared library (dll) that contains pre-specified static functions.

The following callback functions are available:

extern "C" HRESULT CMICusPrepareSynchronize (CATUnicodeString &sFeedback, CATBoolean &bCancel)

Called after Command is activated but before Synchronize dialog is filled.

`sFeedback`	String can be set and will be piped through to consecutive customizing messages, eg. to suspend the custom actions.
`bCancel`	Return `bCancel==true` to cancel the Command. Customization is responsible for user message.
`HRESULT`	`S_OK` if success `E_FAIL` if failure Failure does not stop processing.

extern "C" HRESULT CMICusPreSynchronize (CATUnicodeString &sFeedback, CATBoolean &bCancel)

Called after Synchronize button is pushed.

`sFeedback`	String is supplied and will be piped through to consecutive customizing messages, eg. to suspend the custom actions.
`bCancel`	Return `bCancel==true` to cancel the action. Customization is responsible for user message. Synchronize button will be in deactivated state.
`HRESULT`	`S_OK` if success `E_FAIL` if failure Failure does not stop processing.

extern "C" HRESULT CMICusPostSynchronize (const CATUnicodeString &sFeedback, const CATBoolean bCompleted)

Called after Synchronize button is pushed.

`sFeedback`	String is supplied.
`bCompleted`	TRUE if Synchronize succeeded
`HRESULT`	`S_OK` if success `E_FAIL` if failure Failure does not stop processing.

A sample Visual Studio Project to create the customizing DLL is provided in data\CMICAA\CMICusCallbackWorkspace.zip

Configuration

The custom callbacks are enabled by setting

CMI_ENABLE_CUSTOMIZATION=ON

CMICusCallback.dll is installed in the %PATH%

CHAPTER 5

Optional CMI-Features

Plotting a model

Conventions

The *CATIA Teamcenter Interface* permits an automatic update of title blocks in CATIA-models with title block values that are retrieved from Teamcenter Enterprise. The CMI functionality identifies title blocks in a CATIA model by means of a naming convention. This naming convention is threefold in the way that it affects three levels of CATIA data in the model whose identifiers have to start with strings qualifying them to be constituents of a title block.

Those qualifying identifier sub strings are to be defined for

...drafts: if any draft identifier **starts** with the qualifying string the CMI function continues to search for

...views: if any view identifier **starts** with the qualifying string the CMI function continues to search for dittos referring to ...

...details: if any detail identifier **starts** with the qualifying string defining a title block the CMI function retrieves all values defined for this title block from Teamcenter Enterprise and displays them in the corresponding view positions.

There is only one (configurable) start string for drafts and one other for views but there may be a multitude of strings for details (as many as different title blocks) defining their corresponding title blocks. Thus the need for probably many different title blocks in a company is covered.

The following figure shows a simple draft with the name conventions used by the *CATIA Teamcenter Interface*.

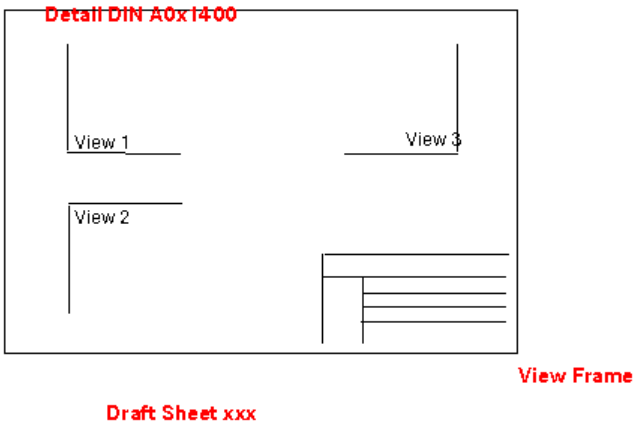

Figure 11: Simple draft with drawing frame conventions

Sample for the definition of some naming conventions

In the following we describe a sample which fits to the conventions that we need to define in the file dshdrawingframe.sh. This file must be located in a path that can be found in $PATH. You can find a sample for this file in your catedm installation path catedm/data/dshdrawingframe.sh.

In this example we will define

Draft : BLATT

View : RAHMEN

Detail : 'DIN A3' and 'DIN A5'

Catedm/data/dshdrawingframe.sh

```
..
CATIAFRAME)
    Case $TOKEN in
      ALL)
        echo "redraw-default yes"
        echo "indicatordraft BLATT"
        echo "indicatorview RAHMEN"
        echo "standard_format DIN"
        echo "frames {\"DIN A3\" \"DIN A5\"}"
        ;;
      redraw-default)
        echo "yes"
        ;;
      indicatordraft)
        echo "BLATT"
        ;;
      indicatorview)
        echo "RAHMEN"
        ;;
      standard_format)
        echo "DIN"
        ;;
      frames)
        echo "{DIN A3} {DIN A0}"
        ;;
      *) ;;
    esac
    ;;
…
```

Creation of a CATIA-Model

Now we create a CATIA-Model with CMI and CATIA that fits to the previously declared naming conventions (see appdefault.sh) :

- Load a CATIA-Model with the CATEDM functionality
  `METAPHSE: READ`

- Split CATIA screen into "DRAFT" and "3D" with
  `"IMAGE → DEFINE"`

- Create a "Draft" with
  `"DRAFT → CREATE"`

  SP (SPace (3D)) -> DR (DRawing) : activate **DR**awing mode
  (name e.g. "Blatt")

- Create geometry (e.g. a "PLANE")
  DR → SP : activate 3D **SP**ace mode

- Create a 2D point in your new created Draft "POINT -> COORD"
  SP → DR : activate **DR**awing mode

- Create "View" with
  "AUXVIEW → CREATE" functionality.

  SP → DR : activate **DR**awing mode
  Describes the projection to the plane created before.

  SELECT LINE/ PLANE (created in point 3)
  SELECT POINT (created in point 4)
  KEY VIEW ID (e.g. "Rahmen")

- Create a Detail with **`"DETAIL → CREATE"`**
  KEY DETAIL ID (name e.g. "DIN A3")
  Select in your 2D View (created with AUXVIEW) a Line/Point to create a Ditto onto this view.

- Save the model
  METAPHSE: UPDATE → ACTIVE

Definition of the relevant attributes

The customizing of the company specific drawing frames now can be made in the shell script *drawingframe* that can be found in directory 'catedm/bin'. In this script file the user can define several views like „**DIN A3**“ whereat a view can contain several entries. An entry for an example looks like the following:

```
{string { "Company Name" }}            {xpos -380} {ypos 0}  {size 10}
{string {Created: @Date}}              {xpos 2}    {ypos 62} {size 1}
{string {Modified: @ModificationDate}} {xpos 2}    {ypos 42} {size 1}
{string {Name: @Name}}            {xpos -0}   {ypos 5}  {size 1.5}
                                       {color 3}
{string {User: $USER}}          {xpos -20}  {ypos 2}  {size 3}
{orientation 1}
```

Each view section in the drawingframe script file can contain the following entries within a single line:

Statement	Syntax	Description
Xpos	{xpos x}	x position of text entry
Ypos	{ypos y}	y position of text entry
Size	{size s}	size of text entry
Color	{color c}	color of text entry
Orientation	{orientation o}	orientation of text entry
String	{string {sub string} }	text entry.

The *sub string* can contain following statements:

Statement	Example	Description
“string“	“T-Systems“	The string will be written into the drawing
String	T-Systems	The string will be written into the drawing
$<var>	$USER	The UNIX user name will be written into the drawing
@<var>	@Date	The alias name for the x3GetAttrsForCAD message. In our example listing it is the alias name for the Teamcenter attribute “CreationDate“.

The parameter that begin with "@" are relevant for the customization. In this case, the parameter list:

Date

ModificationDate

Name

will be transferred to the *x3GetAttrsForCAD* CMI method as *Attrs_demanded* string list. Now this string list can be scanned; for each custom parameter the variable *AttrVals_found* is extended and given back to the CMI server. The CMI server can now actualize the drawing frame with the most recent database information.

Please consult the files

custom/g3Custom.mth (GMI related methods) and custom/x3Custom.mth (CMI related methods) for more information.

LINX customizing methods

From LINX to CATIA

Rebuild data structure from LINX in CMI-WB with calling OMF client and send it automatically to CATIA.

Override:

x0WkBnch:x3FindCatModelByCusKey

x0WkBnch:x3FindCatPartByCusKey

x0WkBnch:x3FindCatRelByCusKey

in x3Custom.mth

Find with custom attributes the OBID from Model

→ x0WkBnch:x3FindCatModelByCusKey in x3Custom.mth

Find with custom attributes the OBID from Part

→ x0WkBnch:x3FindCatPartByCusKey in x3Custom.mth

Find with custom attributes the OBID from Relation

→ x0WkBnch:x3FindCatRelByCusKey in x3Custom.mth

Standard Properties in CATIA V5

In CATIA V5 standard properties (Revision, Definition, Nomenclature and Description) can be set from CMI. The property values could be changed by the user and all changed properties are sent back to Teamcenter Enterprise during update. Standard properties may be set / stored for the Part-, x0CatPrd- or x0CtFile-Class.

Customization: Sending standard attributes to CATIA V5

Overwrite:

x3GetCATIARevision

x3GetCATIANomenclature

x3GetCATIADefinition

x3GetCATIADescription

Customization: Receiving user defined attributes from CATIA V5

It is possible to receive changed standard properties from CATIA V5. These attributes may be saved back in Teamcenter Enterprise.

Overwrite:

x3SetCATIARevision

x3SetCATIANomenclature

x3SetCATIADefinition

x3SetCATIADescription

User Defined Properties in CATIA V5

In CATIA V5 you can add user-defined properties to the standard CATIA V5 properties form (Added Properties).

CMI provides two new messages to work with such user defined properties.

It is possible to send user defined properties from Teamcenter Enterprise to CATIA V5 and display these properties within the standard properties dialog. The property values could be changed by the user and all changed properties are sent back to Teamcenter Enterprise during update.

It is not possible to define new properties in CATIA V5 dialog and save them back to Teamcenter Enterprise.

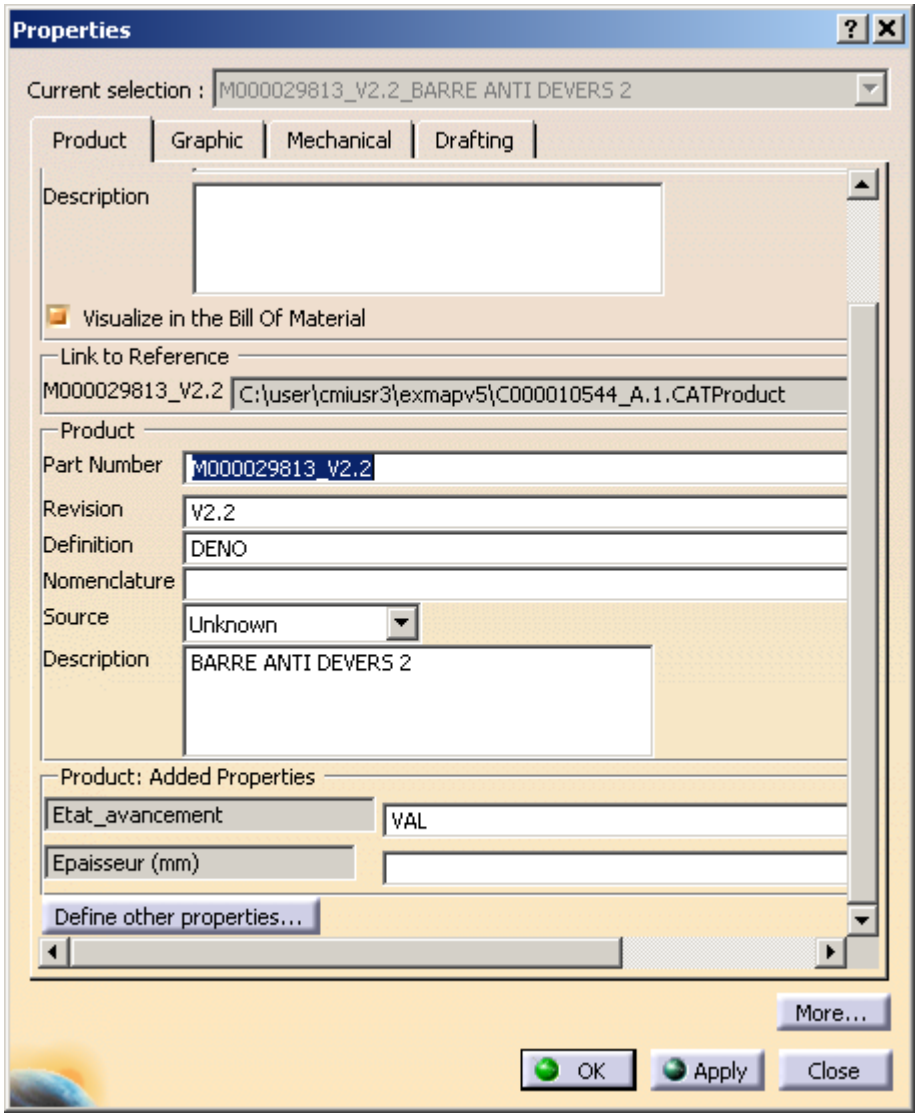

Figure 12: User defined properties

Customization: Sending user defined attributes to CATIA V5

It is possible to send user defined attributes to CATIA V5 and display them within the standard CATIA V5 properties dialog. Therefore you have to customize the CMI message "**x3SendUserDefProps**". This message is attached to several classes → it is possible to add properties for CATProducts, CATParts, V4 models and Black Box objects.

```
x0CTFile/Part/x0CatPrd: x3SendUserDefProps(
   input:       ObjectPtr    thisObj          ::
   output:      SetOfStrings *vPropDisplaySet ::
   output:      SetOfStrings *vPropNameSet    ::
   output:      SetOfStrings *vPropValueSet   ::
   output:      integer *    mfail) code
```

```
vPropDisplaySet … the display names of attributes within
                  properties dialog
vPropNameSet    … the database name of the attributes to be able
                  to save changed values back during update
vPropValueSet   … the attribute values
```

If you want to add user defined properties to the CATProduct properties dialog you have to customize "**Part:x3SendUserDefProps**"

If you want to add user defined properties to the CATPart, V4 Model or Black Box CATPart properties dialog you have to customize "**x0CTFile:x3SendUserDefProps**"

If you want to add user defined properties to the Black Box CATProduct object properties dialog you have to customize "**x0CatPrd:x3SendUserDefProps**"

If you want to delete user defined properties you have to customize the method **x3SendUserDefPropsExt** (Wrapper of **x3SendUserDefPropsExt):**

```
x3SendUserDefPropsExt(
  input:     ObjectPtr    thisObj          ::
  output:    SetOfStrings *vPropDisplaySet ::
  output:    SetOfStrings *vPropNameSet    ::
  output:    SetOfStrings *vPropValueSet   ::
  output:    SetOfStrings *vPropDelSet     ::
  output:    integer *    mfail)
```

where vPropDelSet contains the attribute-names to delete in Catia V5

Customization: Receiving user defined attributes from CATIA V5

It is possible to receive changed user defined properties from CATIA V5; These attributes may be saved back in Teamcenter Enterprise. Therefore you have to customize the CMI message "**x3ReceiveUserDefProps**". If newly created user defined properties within CATIA V5 standard dialog should be stored in Teamcenter Enterprise, they have to be declared in the CMI_CONFIGURATION_FILE.

```
x0CTFile/Part/x0CatPrd: x3ReceiveUserDefProps(
   update:      ObjectPtr    thisObj         ::
   input:       SetOfStrings vPropDisplaySet ::
   input:       SetOfStrings vPropNameSet    ::
   input:       SetOfStrings vPropValueSet   ::
   output:      integer *    mfail) code
```

The declaration in the CMI_CONFIGURATION_FILE of newly created properties in catia, which should be stored in Teamcenter Enterprise looks as follows:

```
<UserDefinedProperties>
            <UserDefinedProperty Name="CustomerProp1" />
            <UserDefinedProperty Name="CustomerProp2" />
</UserDefinedProperties>
```

Customization: using CATIA V5 Properties during Part creation or model registration

The customizing messages *x3CreatePartExt* and *x3CreateFileForUpdtExt* provide property attributes from Catia V5, in order to have this information availble before a Part, Document or Model is created in Teamcenter.

```
class message x0WkBnch:x3CreatePartExt(
   input  :      string     strClassname    ::
   input  :      string     newPartClass    ::
   output :      ObjectPtr *partObject      ::
   input  :      NvSet      pAttributeSet   ::
   input  : NULL string     strPartCategory ::
   input  :      boolean    bUseDialog      ::
   update :      SetOfStrings *vStats       ::
   output :      integer   *mfail) code


message g0GenMod:x3CreateFileForUpdtExt (
   update:      ObjectPtr     thisObj      ::
   input :      string        file         ::
   input :      string        partnumber   ::
   input : NULL NvSet         ModelInfos   ::
   input : NULL NvSet         PartAttributes ::
   input : NULL SetOfStrings  matrix       ::
   update: NULL ObjectPtr     existingPart ::
   output:      ObjectPtr *   createdPart  ::
   output:      ObjectPtr *   document     ::
   output:      ObjectPtr *   relObject    ::
   update: NULL SetOfStrings  vStats       ::
   output:      integer      *mfail) code
```

The NvSet arguments pAttributeSet and Part Attributes respectively contain the following CATIA Standard Properties:

```
CATIADefinition
CATIANomenclature
CATIARevision
CATIADescription
Nomenclature
PartNumber
```

Use nvsGet to retrieve the values of these properties.

The default implementation of these methods can be found in the file x3Custom.mth.

You can retrieve User Defined Properties that were present in CatiaV5 via the function x0WkBnch:x3ConvertUserDefProps

```
dstat= x3ConvertUserDefProps(x0WkBnchClass,
                             ModelInfos,
                             &vDisplayNameSet,
                             &vPropNameSet,
                             &vPropValueSet,
                             mfail);
```

for Model-Properties or

```
dstat= x3ConvertUserDefProps(x0WkBnchClass,
                             PartAttributes,
                             &vDisplayNameSet,
                             &vPropNameSet,
                             &vPropValueSet,
                             mfail);
```

for Part-Properties.

Configurable Behaviors in CATIA V5

It is possible to configure the behavior of catia while update & synchronize. Dependend on a given prefix of the partnumber in a CATProduct/Component it is possible to force the update to:

- Ignore a Component.
- Ignore the CATProduct/Component and it's subtree.
- Create a special relation in Teamcenter

This has to be configured in the CMI_CONFIGURATION_FILE. The scheme looks as follows:

<CMIConfigTopics>

.

.

```
<ConfigurableBehaviors>
      <ConfigurableBehavior UniqueID = "[Unique ID]">
            <BehaviorType>[Behavior Type]</BehaviorType>
            <PartNumberPrefix>[Prefix]</PartNumberPrefix>
            <Behavior>[Behavior]</Behavior>
      </ConfigurableBehavior>
</ConfigurableBehaviors>
.
.
</CMIConfigTopics>
```

There can be multiple tags **<ConfigurableBehavior>** in the tag **<ConfigurableBehaviors>**

The tag **<UniqueID>** has to contain a value which have to be unique in this file.

The following options exist to define which CATIA components a configurable behavior shall apply to:

Based on its part number -

<PartNumberPrefix>Spec_**</PartNumberPrefix>**

The behavior applies to components whose part number begins with “Spec_”

An empty PartNumberPrefix makes the behavior apply to any component.

Based on its product type -

<ProductType>ElecWireGroup**</ProductType>**

The behavior applies to components of the type ElecWireGroup. To help with configuration, the Product type is shown in the CMI Info dialog.

Based on its instance name -

<InstanceNamePrefix>XY_**</InstanceNamePrefix>**

The behavior applies to components where the instance name begins with “XY_”.

When these tags are combined, a component must match the requirement of either tag.

The following combinations of values are valid:

No.	BehaviorType	Behavior
1	EmbeddedNodeBehavior	SkipNode
2	EmbeddedNodeBehavior	DeepSkipNode
3	EmbeddedNodeBehavior	IgnoreNode
4	EmbeddedNodeBehavior	ReferenceGeometry
5	ProductNodeBehavior	IgnoreNode

Descriptions of the behaviors:

1. **SkipNode:**

The component is skipped and its children are instantiated as a direct child of the CATProduct/Assembly that contains the component.

2. **DeepSkipNode:**

The component and all subsequent components beneath it are skipped, up to the next regular CATProduct/CATPart.

3. **IgnoreNode:**

The Component and its substructure is ignored in Teamcenter.

4. **ReferenceGeometry**:

The Component is skipped and its children are instantiated in Teamcenter with a special Reference relation instead of the standard Assembly relation. So the substructure of this component will not be part of the BOM.

5**. IgnoreNode (Product):**

CATProduct is ignored in Teamcenter. This may result in broken links as the Product is not provided by Teamcenter during a load.

Usage of database name of objects

There is the possibility to query for objects by OBID and DB name during a CMI update action instead of only querying by OBID in all databases in scope or session.

To enable this feature you have to set the following configuration variable within PDM_CONFIG (config.cfg):

CMI_USE_DB_NAME = "ON"

During the CMI Read action the database names of all objects will be read and written to the Object Manager file. During update it is now possible to query for objects exactly within the database they are saved in.

Additionally you may customize some or all of the following messages to your own needs:

x3GetRelevRelViaObidDb

x3CusGetRlvRlViaObidDb

x3GetPartByAttrsAndDb

x3GetModPosViaObidDb

x3GetPrdPartViaObidDb

You can find these messages within x3Custom.mth

Customization messages

Messages contain input parameter database name which is the Teamcenter Enterprise attribute "CurDbName" of the object.

Rest of code should be the same as in already existing messages "Get…ViaObid". But instead of "QueryWhere" or "QueryDbObject" the database specific message QueryWhereByDbName" is used.

```
class message x0WkBnch:x3GetRelevRelViaObidDb (
   input :       string    ClassName    ::
   input :       string    RelOBID      ::
   input : NULL  string    DataBaseName ::
   output:       ObjectPtr *Relation    ::
```

```
   output:      integer   *mfail) code

class message x0WkBnch:x3CusGetRlvRlViaObidDb (
   input :      string    ClassName   ::
   input :      string    RelOBID     ::
   input : NULL string    RelDbName   ::
   input : NULL ObjectPtr ctItemObj   ::
   output:      ObjectPtr *Relation   ::
   output:      integer   *mfail) code

class message x0WkBnch:x3GetPartByAttrsAndDb (
   input :      string      ClassName     ::
   input : NULL string      databaseName  ::
   input : NULL string      partOBID      ::
   input : NULL string      partNumber    ::
   input : NULL string      partRevision  ::
   input :      string      lastSeq       ::
   output:      ObjectPtr   *subassObj    ::
   output:      integer     *mfail) code

class message x0WkBnch:x3GetModPosViaObidDb (
   input :      string       className     ::
   input :      string       ModPosClass   ::
   input :      string       Obid          ::
   input : NULL string       DataBaseName  ::
   output:      ObjectPtr    *ModPosObj    ::
   output:      integer      *mfail) code

message ProdBI:x3GetPrdPartViaObidDb(
   input :      string       className       ::
   input :      string       partOBID        ::
   input : NULL string       partDbName      ::
   input : NULL NvSet        customData      ::
   output:      ObjectPtr    *partObj        ::
   output:      integer      *mfail) code
```

Read Reference/Sheet Information from CATIA V5 Drawings

Reference Documents for Drawings

Via the CATIA V5 config-variable **`CMI_REFERENCE_OF_DRAWING = "ON"`** all "Reference documents" of a CATDrawing are stored in the CatDrawing-model-info of the exchange-file. You can use these links to attach the drawing to a relevant folder. Customization can get this information with the method x3SetCusAttrModInf. Within this method you have to call:

"dstat= nvsGet(ModelInfos, " SOURCE_PARTNUMBERS ", &ModelData);" to get the reference part numbers.

"dstat= nvsGet(ModelInfos, " SOURCE_DOCS ", &ModelData);" to get the reference documents.

"dstat= nvsGet(ModelInfos, " SOURCE_OBIDS ", &ModelData);" to get the reference part OBIDS.

Sheet-Information for Drawings

`Set "CMI_CALC_SHEETS=ON`" in the CATIA V5 Environment.

Use "dstat= nvsGet(ModelInfos, "SHEETS", &ModelData);" to get all information on the sheets.

Enhanced 4D-Navigator Integration

New custom-point: **`x3Inflate4DNavName`**

Handles the x3InflateExMapName message for the x0CTFile class. Message type is an Object Message. File system name within exchange map is identical to the RelativePath of the x0CTFile object.

As standard the RelativePathAttr is used.

New custom-point: **`x3GetSceneFile`**

This method interacts with the user to get the Scene file describing the situation in the 4D-Navigator In the parameter fileContents the content of the file is handled back to the calling method.

Notice: You can also load the Scene file from a pre-defined directory (like the Scene file creation)

As standard the 4D-Nav-Exchangemap is used as relative path, with an attached "/NEW_SCENE.wrl".

CATIA project environment support

New custom point **`x3SetCatiaProject`**.

Handles the x3SetCatiaProject message for the x0CTFile class.

Set a new attribute to a Catia model, when it's updated, created and saved in Catia.

Support of Teamcenter "Quantity"

In standard Teamcenter Enterprise, the "AssmStrc" relation class carries the attribute "Quantity". A part can be used multiple times in an assembly, according to the value set for the "Quantity". In order to support the multiple usage of parts within an assembly, the "x2AsmPoQ" relationship carries a number of transformation matrixes. The picture below shows the extensions within the data model:

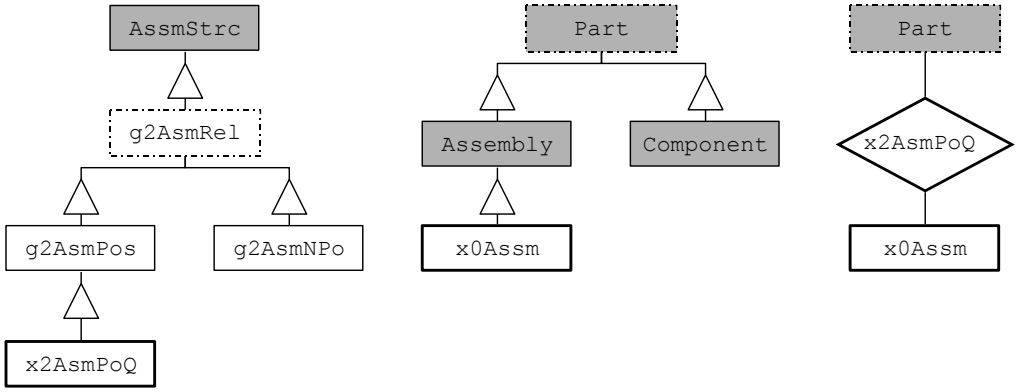

Figure 13: Data model extensions for support of 'Quantity'

Customizations may utilize the *x0Assm* class in order to create *x2AsmPoQ* relationships. The *x2AsmPoQ* relationship stores the amount of 'Quantity' tranformation matrices within a 'FullTable' attribute *x0TrafoList*. Each single transformation matrix can be identified by an index passed to the CMI APIs *g3GetTrafo* and *g3UpdateTrafo* (see description below).

In order to ensure the consistency between the value of the 'Quantity' attribute and the number of transformation matrices, the 'Quantity' mustn't be edited manually! This means that all functionality like 'Update Relationship' e.g. has to be altered in a way which prevents the user from editing the 'Quantity' attribute. If the 'Quantity' is raised, a new unity matrix is added by default.

For each item of Quantity an instance of the Model geometry will be shown in CATIA, and a transformation matrix has to be kept in the database. Therefore you should limit the Quantity to a reasonable number, eg. in user dialog validation. In the standard implementation of CMI this is controlled by setting the variable *GCVMI_MAX_QUANTITY* in your *config.cfg* file. If the variable is not set, there is a limit of 100 for the Quantity attribute.

Example:

SET GCVMI_MAX_QUANTITY "20";

Working with CATIA V5 Released Cache

CMI supports the use of CGR-files in the released cache of CATIA V5. For this purpose the Teamcenter customization has to store the CGR-files of Catia-models in a specific CGR-Vault. During "To Catia" these CGR-files are copied to the Released Cache instead of the Catia-models to the exchange-map. In CATIA V5 the CGR-files are loaded in visualization mode. For each file in the workbench CMI decides via customizable methods whether to copy the standard file or the CGR-file.

Options -> Change Preferences -> CMI Preference.

Set “Transfer CGR-File to CATIA V5” to “Only CGR”/ “CGR + geometry”

In CATIA you have to use the following settings

- Work with the cache system = ON
- Path to the local cache = <path to local cache>
- Path to the released cache = <at least one path to released cache>
- Check timestamps = ON

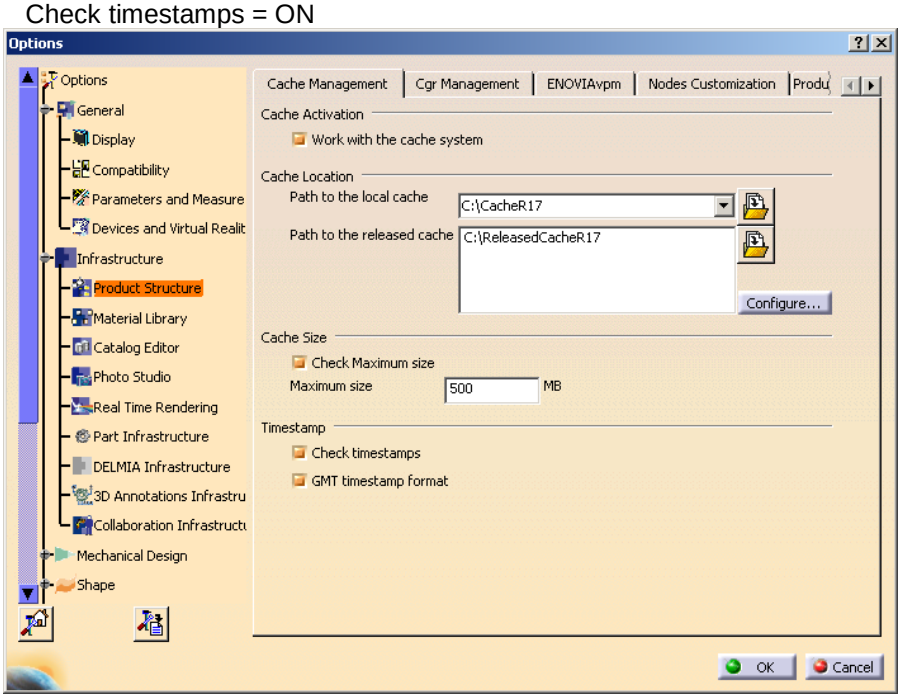

Figure 14: CATIA Options - Cache Management

These are the necessary preconditions to copy a CGR-file:

Work with the cache system in CATIA V5 is enabled.

The Released cache is set and exists.

The CMI-preference “Transfer CGR-File to CATIA V5” is set to “Only CGR” or “CGR + geometry”.

The Catia-model is checked in.

In the CGR-Vault exists the CGR-file with following name: “<model-name>.cgr” where <model-name> is the filename of the Catia-model.

Customization may change the behavior of the ”CGR-search”. Then you should customize the methods x3UseCgrFile and x3GetCgrFilePaths. x3UseCgrFile decides whether to use the CGR-file; x3GetCgrFilePaths delivers the relevant file-path of the CGR-file.

If the customization needs a Teamcenter-class to register the CGR-file, it is recommended to use a subclass of the CMI-class x0V5CGR.("CATIA V5 CGR for Cache File").

Configuration:

1) Settings in $PDM_CONFIG:

```
# host where the CGR files are located (optional)
# if not set CGR files has to be located in the same vault location like the
# corresponding CATPart
set CMI_CGR_HOST "<CGR_HOST>";
```

```
# path to the vault location on the CMI_CGR_HOST (optional)
# if not set CGR files has to be located in the same vault location like the
# corresponding CATPart
set CMI_CGR_VAULTLOC_PATH “<Path to the CGR Vault>";
```

2) Settings during startup of CATIA V5

```
# enable CMI released cache functionality
set CMI_USERELEASEDCACHE=ON

# use a special released cache directory out of the list in CATIA
# this setting is optional; if not set use the first released cache in list
set CMI_RELEASEDCACHEDIR=<Path to released cache>

# enable the “Get original Geometry” button
set CMI_ENABLE_CMIGETORIGGEOCMD=ON

# Optional setting
# use temporary CATIA components which contain the related CGR as shape
representation,
# instead of the original CATPart. (see load structure to CATIA)
CMI_CREATETEMPCGRCOMP=ON
```

3) Settings in CATIA no longer needed to work with released Cache

Earlier releases of CMI didn’t support the CATIA setting “Check Timestamp ON”. So it was necessary to remove the CGR files of the released Cache dir to make sure that the correct version was loaded. Since CMI 8.9 the setting CMI_CLEANRELEASEDCACHE=ON is no longer needed and should not be used.

CATProcess customization

customization methods in TeamCenter

The message g3IsProcessElement is used to detect a process element. The method is called after a drop of a Part in the Workbench. The default method expands to an x0CatPrc data item which is the storage class for a CATProcess file.

```
class message  Part:g3IsProcessElement(
   input : ObjectPtr    thisObj             ::
   input : ObjectPtr    WorkBnchObj         ::
   output: boolean      *bIsProcessElement  ::
   output: integer      *mfail) code
```

The message g3IsProductView is called if a process element expands its children. The Catia V5 structure under the product view is added in the product view of the CATProcess in CATIA V5.

```
class message  Part:g3IsProductView(
   input : ObjectPtr    thisObj             ::
   input : ObjectPtr    WorkBnchObj         ::
   output: boolean      *bIsProductView     ::
   output: integer      *mfail) code
```

The message g3IsResourceView is called if a process element expands its children. The Catia V5 structure under the resource view is added in the resource view of the CATProcess in CATIA V5.

```
class message  Part:g3IsResourceView(
   input : ObjectPtr    thisObj             ::
   input : ObjectPtr    WorkBnchObj         ::
   output: boolean      *bIsResourceView    ::
   output: integer      *mfail) code
```

The message g3GetProcessFiles expands the Document and returns the Process data items, attached to the Document.

```
message ProdBI:g3GetProcessFiles (
   input :      ObjectPtr    thisPtr          ::
   input :      ObjectPtr    WorkBnchObj      ::
   update:      SetOfObjects *processesOfPart ::
   output:      integer      *mfail) code
```

Viewer support

Prerequisites

A custom service has to provide viewing files like JT or CGR. The viewing files are needed for the following geometry types:

CATPart

model

CGR (if you use JT)

CMIArchive (You need one viewing file for the contend of the CMIArchive)

To create the viewing files you can use the following utilities:

CGR: DassaultSystems - CATDMUUtil (Part of CATIA V5)
JT: T-Systems - ComFox Translator

CMI doesn't use viewing files for the CATProducts in the Assembly structure. The structure is created on the fly.

Features

This functionality reads the CMI Workbench content, gets the viewing files (from customisation) and builds up a format file to open viewer.
The CMI Model Filter is extended with a Viewing File type (**`CmiCatiaV5ViewType`**) which will be filtered out by default even if the "Model filter preferences" in the CMI Workbench Window Options menu is not set.

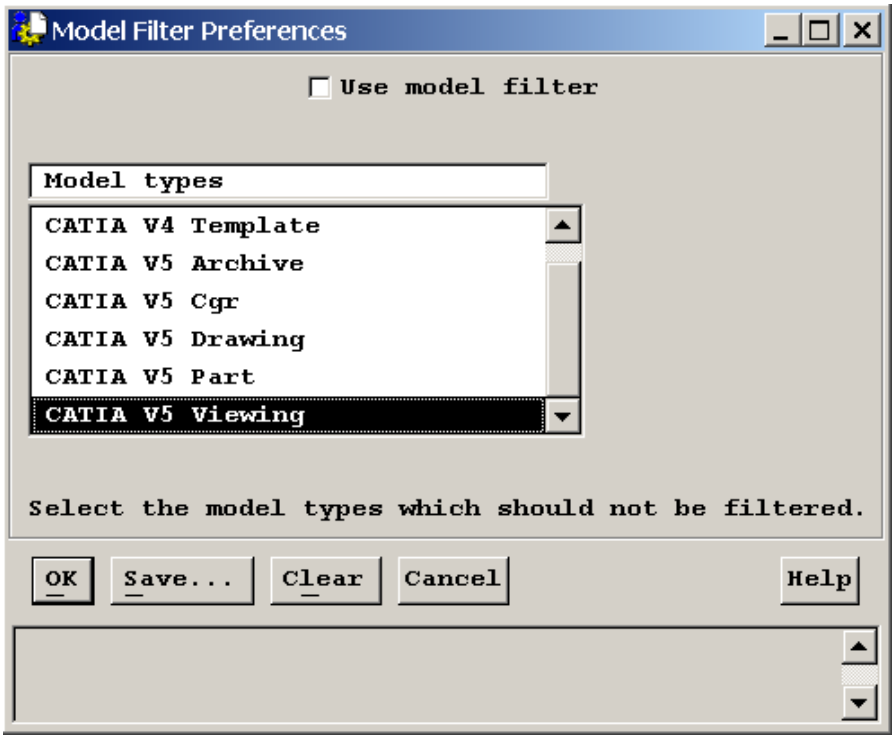

Figure 15: Model Filter Preferences dialog

Configuration

The directory and the viewer format (3dxml, pdmxml, etc.) are configurable in CMI Viewer Preferences.
New Viewer Configuration in CMI Workbench window Options->Change Preferences Menu option.

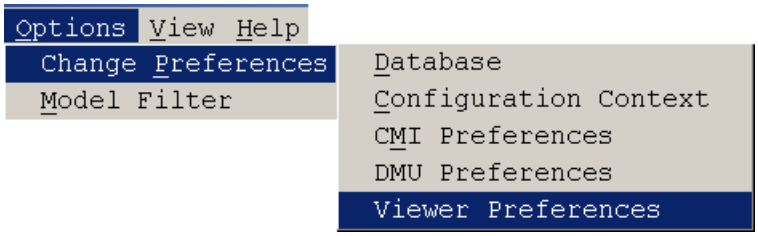

Figure 16: Change Viewer Preference

The Viewer Map directory defines the local Directory which is used for the storage of the viewer file and the related viewing files.

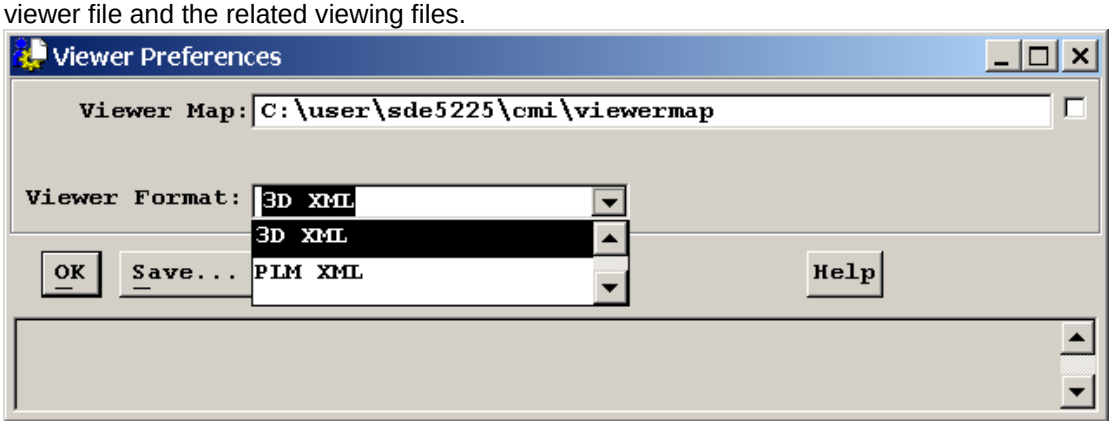

Figure 17: Viewer Preferences dialog

In $PDM_CONFIG file there are the following configuration vars:

If a start script is used to launch the viewer on Windows, the variable `CMI_3DXML_STARTUP_SYNC_WIN` must set to `ON`. This forces the synchronous start of the viewer batch command in Synchronous mode because in the asynchronous mode the execution of .BAT files fail on Windows.

For 3DXML with IE and plug-in:
set **CMI_3DXML_STARTUP** "<path>\StartIE.bat";
set CMI_3DXML_STARTUP_SYNC_WIN "ON";
or for the direct call of the executable:
set **CMI_3DXML_STARTUP** "<path>\code\bin\3DXMLPlayer.exe";

For PLMXML:
set **CMI_PLMXML_STARTUP** "<path>\JT2 GoLauncher.exe";
set CMI_PLMXML_STARTUP_SYNC_WIN "ON";
for batch startup on Windows
set CMI_VIEWER_ENCODING to the charset used in th plmxml – file,
if it is different from “ISO-8859-1”.

Class Constants in x0WkBnch

x4ViewDocToJtRelClassC	= Attach;
x4ViewDocToJtRelShipC	= DataItemsAttachedToBusItem;
x4ViewModToJtRelClassC	= GenDeriv;
x4ViewModToJtRelShipC	= GenDerivSourceOfItem;
x4ViewJtModelClassC	= IndepBin;

Customization

You can define the way JTs or CGRs are retrieved by overriding the following customisation messages.
For JTs attached to the Parts Document, this is:

```
object message x0CTItem:x3GetViewerFileSet (
      input  :     ObjectPtr     partRep        ::
      input  :     ObjectPtr     partObj        ::
      input  :     string        viewFormat     ::
      input  :     ObjectPtr     workBench      ::
      output :     SetOfStrings  *pHosts        ::
      output :     SetOfStrings  *pUsers        ::
      output :     SetOfStrings  *pFullPaths    ::
      output :     SetOfStrings  *pNewFileNames ::
      output :     integer       *iStatus       ::
      output :     integer       *mfail
      )code
```

More than one JTs can be attached to a document.

If the JTs are linked to the CATParts (Has Visualization), it is:

```
message x0ModRep:x3GetViewerFile(
      input :      ObjectPtr   modelRep        ::
      input :      ObjectPtr   modelObj        ::
      input :      string      viewFormat      ::
      input :      ObjectPtr   workBench       ::
      output:      string      *sHost          ::
      output:      string      *sUser          ::
      output:      string      *sFullPath      ::
      output:      string      *sNewFileName   ::
      output:      integer     *iStatus         ::
      output:      integer     *mfail) code
```

Only one JT is expected for any one CATPart.

PartRep/modelRep is the Representation Object for the Object message.
PartObj/modelObj is the related Persistent object.

viewFormat is the viewing format from the Preference, by default **x13DXml (3D XML)** and **x1PlmXml (PLM XML)** are supported by CMI.
workBench is the Workbench Object.
the Return values are sHost, sUser, sFullPath, which defines the Source location of the viewer file.
sNewFileName is the File name to be used in the viewer map.
Return of iStatus == 1 in **x0CTItem:x3GetViewerFile** defines, not to search the models for Representation.

Standard behaviour in CMI for **x0CTItem:x3GetViewerFile:**
3DXML:
Get all documents, then get all models, the first model with the type **CmiCatiaV5ViewType** (Should be CGR) will be returned. `iStatus` will return 1, if cgr is found, →no models will be searched.
PLMXML:
Get all documents then expand through x4ViewDocToJtRelClassC **and** x4ViewDocToJtRelShipC **to** x4ViewJtModelClassC.

Standard behaviour in CMI for **x0ModRep:x3GetViewerFile:**
3DXML:
Get the Cgr in the same manner as the Released Cache Concept from **CMI_CGR_VAULTLOC_PATH** and **CMI_CGR_HOST**.

PLMXML:
Expand through `x4ViewModToJtRelClassC` **and** `x4ViewModToJtRelShipC` **to** `x4ViewJtModelClassC`.

Display User Data

The following methods can be overridden to display custom Teamcenter properties in VisView/VisMockup:

For a Representation / JT:

```
g0RepItm:x3GetInfoForViewFile(
    input :      ObjectPtr   thisRep                  ::
    input :      ObjectPtr   thisObj                  ::
    input :      string      FileName                 ::
    output:      NVSET       *nvAdditionalAttributes  ::
    output:      integer     *mfail)
```

For a Part:

```
x0CTItem:x3GetInfoForViewPart(
    input :      ObjectPtr   partRep                  ::
    input :      ObjectPtr   partObj                  ::
    output:      NVSET       *nvAdditionalAttributes ::
    output:      integer     *mfail)
```

The properties added to `nvsAdditionalAttributes` will be shown as usr data/user values in VisView.

Design Table Support

Features

Design Tables are managed in Teamcenter by Synchronize of Catia V5.

Configuration

In `$PDM_CONFIG` file there is the following configuration var:
`CMI_DESIGN_TABLES`

If this variable is set to “ON”, during *To Catia* all Catia Model/Products are expanded for Design Tables and the relevant Design Tables are transferred to Catia V5. There is a performance impact.

Class Constants

`x0WkBnch.x4CreateCatTxtC = x0CatTxt;` Class for csv design tables

`x0WkBnch.x4CreateCatExcC = x0CatExc;` Class for Excel™ design tables

`x0WkBnch.x4DesTabRelClassC = x2DepDes;` Dependency relation

`x0WkBnch.x4DesTabRelShipClassC = DesTabDependentOnItems;` Relationship

Customizable Methods

```
define message x3GetDesignTables (
  input  :      ObjectPtr    catiaObj       ::
  input  : NULL ObjectPtr    partObj        ::
  output :      SetOfObjects *desTabObjs    ::
  output :      SetOfStrings *dtFileNames   ::
  output :      integer      *mfail);

define message x3IsLinkedWithDesTab (
  input  :      ObjectPtr    modelObj       ::
  input  :      ObjectPtr    desTabObj      ::
  output :      integer      *bLinked       ::
  output :      integer      *mfail);

define message x3LinkWithDesTab (
  input  :      ObjectPtr    catiaObj       ::
  input  : NULL NvSet        ModelInfos     ::
  input  :      ObjectPtr    desTabObj      ::
  input  : NULL string       sPartClassName ::
  input  : NULL string       sPartOBID      ::
  input  : NULL string       sPartDbName    ::
  output :      ObjectPtr    *relObject     ::
  update :      SetOfStrings *vStats        ::
  output :      integer      *mfail);

define message x3RemoveDesTab (
  input  :      ObjectPtr    modelObj       ::
  input  : NULL NvSet        ModelInfos     ::
  input  :      ObjectPtr    desTabObj      ::
  input  : NULL string       sPartClassName ::
  input  : NULL string       sPartOBID      ::
  input  : NULL string       sPartDbName    ::
  update :      SetOfStrings *vStats        ::
  output :      integer      *mfail);
```

These methods are used to get related Design Tables for a Catia Model/Product or to relate a Design Table to a Catia Model/Product.

MML Support

Features

MMLs can be managed in Teamcenmter.

Installation

This use case requires additional CMI CATIA library:

Windows:	CMIExt.dll
Solaris:	libCMIExt.so
HP-UX:	libCMIExt.sl
AIX:	libCMIExt.a
IRIX:	libCMIExt.so

This library is not part of the CMI product. Please contact *cmi_support@t-systems.com* to get more information.
Copy this library to the binary path of the CMI CATIA installation, e.g.
<CMI_Installation_Dir>\CMICATV5_R16_91V10\intel_a\code\bin\

Configuration

Teamcenter setting in $PDM_CONFIG:
Optional:
CMI_DEPENDENT_TYPES
Default: CCP
Set this environment if you want to use other link types than CCP.

Settings during startup of CATIA V5
CMI_GETPOINTEDDOCUMENTS=ON
Default: OFF
Set this environment to “ON” to provide the information about referenced CATIA files to Teamcenter.
CMI_ENABLE_CMIEXTERNALDOCCMD=ON
Default: OFF
Set this environment to “ON” to enable the “Get Referenced Geometries” button in CATIA V5.
CMI_ENABLE_CMIGETDEPBYCMD=ON
Default=OFF
Set this environment to “ON” to enable the “Get Depended By Geometries” button in CATIA V5 (Teamcenter Customizing is required due to version ambiguities.)

Figure 18: MML support buttons

Customizable Methods

Manage dependencies during Update / Synchronize / Create / SaveAs:

```
message x0CTFile:x3UpdateDepV5 (
   input :      ObjectPtr      this          ::
   input : NULL NvSet          ModelInfos    ::
   update:      SetOfStrings   *vStats       ::
   output:      integer        *mfail);
```

Get Referenced Geometries:

```
message x0CTFile:x3DependsOnFiles (
  input  :      ObjectPtr      this          ::
  output :      SetOfObjects   *depFiles     ::
  output :      integer        *mfail);
```

Get Depended By Geometries:

```
message x0CTFile:x3IsDependedOnByFiles (
  input  :      ObjectPtr      this          ::
  output :      SetOfObjects   *depOnByFiles ::
  output :      integer        *mfail);
```

When you are using “Get Depended By Geometries “, you have to make sure your customizing of x3IsDependedOnByFiles filters the depended on by files. It is not possible to load multiple versions of one file into CATIA V5.

Representation Formats in CATIA V5

The CATIA V5 module can be configured to support CATIA V5 shape representation files in a generic way. CATIA V5 Representation files can be used in a product structure and are handled in a similar fashion to V4 Models.

To enable support for specific Shape Representations, an administrator can set the following variable in the Catia V5 environment:

```
CMI_REP_FORMATS={3dmap}{CATShape}
```

The legitimate types can be found in the Manage Representrations dialog in CATIAV5 (in this example 3dmap and CATShape formats will be handled by the CMI Catia Module.)

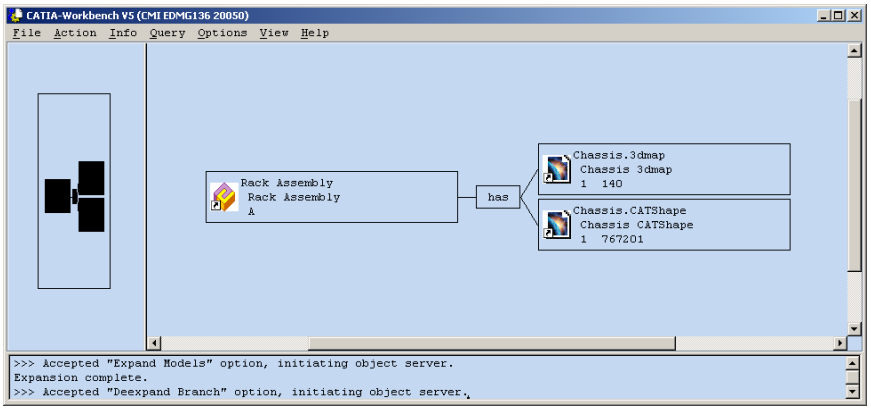

Figure 19: Representation formats in CATIA V5 (CATIA Workbench)

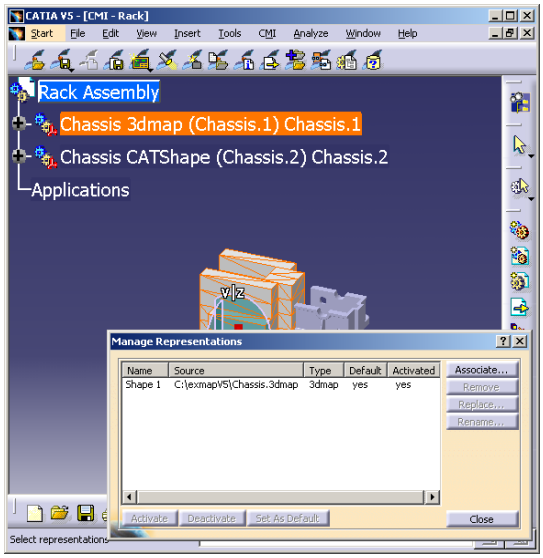

Figure 20: Representation formats in CATIA V5 (CATIA V5)

```
class message x0WkBnch:x3GetClassForCatRep(
   input :         string         sClassName             ::
   input :         string         sOrigCatRepClassName   ::
   input : NULL    string         sOrigFileName          ::
   input : NULL    string         sOrigPartNumber        ::
   output:         string         *sNewCatRepClassName   ::
   output:         integer        *mfail) code
```

Override this message to provide dedicated classes for individual Catia V5 Shape representations. The default Class name returned is x0CatRep. If you introduce your own Representation classes, they should derive from this base class.

Automatic update of CATDrawing title blocks with Teamcenter data

Customizing option to fill in Drawing title block from Teamcenter data. Implement the customizing message `x3GetParamForFileV5` to define Teamcenter data that should be transfered to Knowledgeware Parameters in Catia V5.

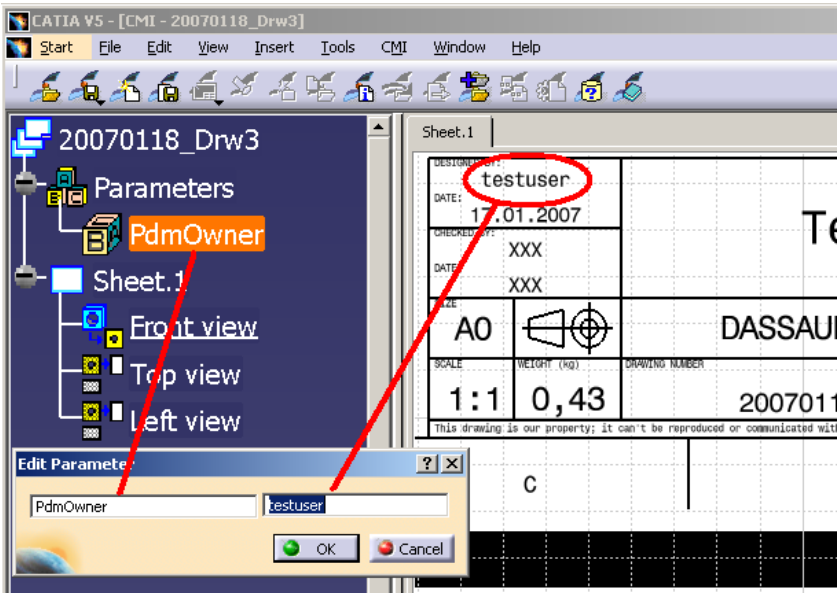

Figure 21: CATDrawing title block

Override the Customizing message `x3GetParamForFileV5` in Teamcenter to define the attributes for Catia:

```
class message g0GenBin:x3GetParamForFileV5(
   input : string         sClassname      ::
   input : ObjectPtr      thisObj         ::
   output: SetOfStrings   *vAttributeList ::
   output: SetOfStrings   *vValueList     ::
   output: integer        *mfail);
```

`sClassname` is the class name of the Drawing object.
`thisObj` is the object itself
`vAttributeList` is the list of parameter names
`vValueList` is the list of related values for the attributes. Both lists must be the same length.

Transfer of weight properties (inertia) from Catia V5 to Teamcenter

Inertia properties – eg. Mass – can be read from Catia V5 and stored in Teamcenter.

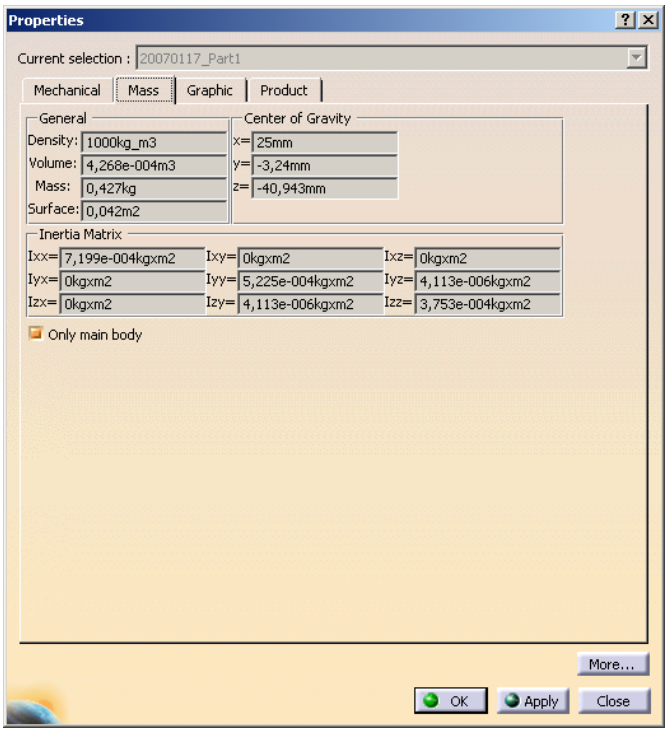

Figure 22: Properties with inertia

In order to read these properties from Catia, the following variable has to be set in the Catia V5 environment:

`set CMI_READ_INERTIA=ON` (Inertias from CATProducts and CATParts are sent to Teamcenter.)

`set CMI_READ_INERTIA=ONLY_CATPART` (Only the inertias from the CATParts will sent to Catia. This may improve the performance especially in case of large assemblies.)

The inertias can confined to the main bodies of the CATParts by a further configuration variable:

`set CMI_CONFINE_INERTIA_TO_MAINBODIES=ON`

The properties can be used in Teamcenter by overriding the customizing method `g0GenBin:x3SetCusAttrModInf` for the CATProduct and/or the CATPart. The customizing method is called in the course of Update, Create and similar actions. CMI itself does not use this information, but you can store it in custom attributes.

```
message g0GenBin:x3SetCusAttrModInf (
   update: ObjectPtr  this        ::
   input : NvSet      ModelInfos  ::
   output: integer    *mfail) code
```

The following named values are provided to the API in the NvSet ModelInfos:

mass
`INERTIA_MASS`

position of the center of gravity
`INERTIA_POSITION_0`
`INERTIA_POSITION_1`
`INERTIA_POSITION_2`

inertia matrix
`INERTIA_MATRIX_0`
`INERTIA_MATRIX_1`
`INERTIA_MATRIX_2`
`INERTIA_MATRIX_3`
`INERTIA_MATRIX_4`
`INERTIA_MATRIX_5`
`INERTIA_MATRIX_6`
`INERTIA_MATRIX_7`
`INERTIA_MATRIX_8`

components of principal axes
`INERTIA_COMPONENTS_0`
`INERTIA_COMPONENTS_1`
`INERTIA_COMPONENTS_2`

`INERTIA_COMPONENTS_3`
`INERTIA_COMPONENTS_4`
`INERTIA_COMPONENTS_5`
`INERTIA_COMPONENTS_6`
`INERTIA_COMPONENTS_7`
`INERTIA_COMPONENTS_8`

principal moments values
`{INERTIA_VALUES_0} 1.0341075818066402e-004`
`{INERTIA_VALUES_1} 1.0341075818066402e-004`
`{INERTIA_VALUES_2} 1.9634954084936208e-004`

`{INERTIA_VOLUME} 1.5707963267948968e-004`
`{INERTIA_DENSITY} 1.0000000000000000e+003`
`{INERTIA_AREA} 2.1991148575128551e-002`

Set Bom-Type of new Catia-files by Teamcenter-customization

In the Synchronize-dialog all new Catia-files are presented with a Bom-Type(Bom/Non-Bom/Not Set). This Bom-Type is initialized by a default and can be changed by the user. Depending on the Catia-V5-config-variable CMI_GET_BOMTYPE_FROM_TC = ON the values for the Bom-Type are fetched from Teamcenter.

With Standard-CMI the bomType is set to “”, i.e. the Bom-Type in Catia remains unchanged.

If you want to customize the setting of the Bom-Type you have to override:

```
message x0WkBnch:x3GetBomType(
   input:       string        strClassname        ::
   input:       NVSET         nvsObjInfos         ::
   input:  NULL NVSET         nvsUserDefInfos     ::
   input:  NULL NVSET         nvsParentObjInfos   ::
   input:  NULL NVSET         nvsParentUserDefInfos::
   output:      string        *bomType             ::
   output:      integer *     mfail) code
```

Allowed values for the return-value bomType:
- “NOT_SET”
- “BOM”
- “NOT_BOM”

For all other values the Bom-Type in Catia remains unchanged.

The `NVSET nvsObjInfos` contains the following Catia attributes for the new Catia-File, if available:

```
PARTNUMBER
FILENAME
NOMENCLATURE
```

The `NVSET nvsUserDefInfos` contains the User-Defined-attributes for the new Catia-File, if existent.

The `NVSET nvsParentObjInfos` contains the following attributes for the Father-Object, ie. the Assembly, if existent:

```
PARTNUMBER
NOMENCLATURE
DB_NAME
OBID
CLASSNAME
```

The `NVSET nvsParentUserDefInfos` contains the User-Defined-attributes for the Father-Object, if existent.

Custom Expand in the CMI CATIA Workbench

The method for the multi-level expand in the CMI CATIA workbench was extended by an optional expand information string. If no special expand information is given, then the default expansion is being performed. Giving an expand information string to the new multi-level expand method will pass this string to the lower-level custom methods that define which sub-parts should be returned for a given assembly.

With this CMI extension customization can define one or more new "custom expand" context menu options in the CMI CATIA workbench context menu. The custom methods that are called by these menu options can call the extended CMI multi-level expand method, passing it a specific expand information string.

The expand information string is stored in the CMI-Filter-preference. At the end of the command this attribute is reset in the filter-preference.

If you want to customize the Expand of an Assembly-structure with the catia-models in the WorkBench you have to customize the following methods:

define trusted external nondisplaying nonprompting class message
```
g3GetAllModelsOfDocCus (
   input :      string       classname  ::
   input : NULL string       expandInfo ::
   input :      ObjectPtr    this       ::
   input :      ObjectPtr    WorkBnchObj::
   output:      SetOfObjects *modelSet  ::
   output:      SetOfObjects *posSet    ::
   output:      integer      *mfail);
```

`g3GetAllModelsOfDocCus` delivers all models and trafo-relations(if existent) of a given folder-object. Depending on the expandInfo, you can call your own method.

```
define trusted external class message g3GetPartsInAssSetCus (
   input :      string       classname  ::
   input : NULL string       expandInfo ::
   input :      SetOfObjects assObjSet  ::
   output:      SetOfObjects *partsInAss::
   output:      SetOfObjects *partRels  ::
   output:      integer      *mfail);
```

`g3GetPartsInAssSetCus` delivers all child-parts and AssmStrc-relations for a given part-object. Depending on the expandInfo, you can call your own method. This method is called for the setbased expand (standard CMI-behaviour, g3SetBasedAllowed = TRUE)

```
define trusted external nondisplaying nonprompting class message
g3GetPartsInAssemblyCu (
   input :      string       classname  ::
   input : NULL string       expandInfo ::
   input :      ObjectPtr    assObj     ::
   output:      SetOfObjects *partsInAss::
   output:      SetOfObjects *partRels  ::
   output:      integer      *mfail);
```

`g3GetPartsInAssemblyCu` delivers all child-parts and AssmStrc-relations for a given part-object. Depending on the expandInfo, you can call your own method. This method is called for the non-setbased expand(g3SetBasedAllowed = FALSE).

Customization example

You have a new menu item *MyMultiLevelExpand* and you want to filter the assembly-structure by your own behaviour:

Define the method and attach it.

```
define message MyMultiLevelExpand like SelectedItemMsg;

attach object message MyMultiLevelExpand to g0GenItm in server
cussvr;
```

Attach your custom expand to the popup menu of the Assembly in the Workbench, below the default Expand

```
define option MyMultiLevelExpandOpt
   using message MyMultiLevelExpand
   with relation g2GnItDp
   with relationship useslowerGI;
```

```
display MyMultiLevelExpandOpt as "My Expand Multiple Levels";

attach option MyMultiLevelExpandOpt to GITreePopupEiwOptL
after GIuseslowermultiGIEiwOpt;
```

write the following method code:

```
message g0GenItm: MyMultiLevelExpand (
   update: ObjectPtr   this ::
   output: integer     *mfail) code
{
   string expandInfo = NULL;
   *mfail= USC_OKAY;
   dstat= uiDeactivateWindow ();
   /* ---------------------------------------- */
   /* Call internal part of method             */
   /* ---------------------------------------- */
   string expandInfo = "myPartExpand";
   dstat= g3ExpandBranchMultiInt (this, expandInfo, mfail);
CLEANUP:
EXIT:
   uiActivateBrowserWindow ();
   return (dstat);
}
```

Now override g3GetPartsInAssSetCus to make your custom expand return only a special type of assemblies:

```
O:attach class  message g3GetPartsInAssSetCus to g0GenWB in server
cussvr;

class message g0GenWB:g3GetPartsInAssSetCus (
   input :      string       classname  ::
   input : NULL string       expandInfo ::
   input :      SetOfObjects assObjSet  ::
   output:      SetOfObjects *partsInAss::
   output:      SetOfObjects *partRels  ::
   output:      integer      *mfail) code
{
   *mfail= USC_OKAY;

   dstat= g3GetPartsInAssSet (classname, assObjSet,
                              partsInAss, partRels, mfail);


   if (expandInfo && nlsStrCmp(expandInfo, "myPartExpand") == 0))
   {
      /* ------------------------------------------- */
      /* TODO: Filter partsInAss/partRels            */
      /* according to the special requirements of your  */
      /* custom Expand                               */
      /* ------------------------------------------- */

      ...
   }

   ...
```

Supply Attributes from Catia to Teamcenter and back

CMI provides a set of APIs which allows the transfer of attributes from CATIA to Teamcenter and back. This functions can only be used within a customer specific CATIA application (CAA or Automation / VBA). Customizing on Teamcenter is required.

Supply Attributes from Catia to Teamcenter

CAA-Function:

```
HRESULT CMIGetPartAttributeFromTC(const CATIProduct_var spProduct,
                                  const CATString &sAttribute,
                                  CATString &sValue,
                                  CATString &sReturnCode);
```

Get Attribute from Teamcenter.
This Function calls x3GetAttValueForObject in Teamcenter.

In:	spProduct	CATProduct to examine in Teamcenter
	sAttribute	sAttributeName
Out:	sValue	sAttributeValue
	sReturnCode	Message from Teamcenter

CAA-Function:

```
HRESULT CMIGetDrawingAttributeFromTC(const CATDocument *pDocument,
                                     const CATString &sAttribute,
                                     CATString &sValue,
                                     CATString &sReturnCode);
```

Get Attribute from Teamcenter.
This Function calls x3GetAttValueForObject in Teamcenter.

In:	pDocument	CATDrawing to examine in Teamcenter
	sAttribute	sAttributeName
Out:	sValue	sAttributeValue
	sReturnCode	Message from Teamcenter

Supply Attributes from Teamcenter to Catia

CAA-Function:

```
HRESULT CMISupplyPartAttributeToTC (const CATIProduct_var spProduct,
                                    const CATString &sAttribute,
                                    CATString &sValue,
                                    CATString &sReturnCode);
```

Supply Attribute to Teamcenter:
This Function calls *x3SupplyAttValueForObject* in Teamcenter.

In:	spProduct	CATProduct to examine in Teamcenter
	sAttribute	sAttributeName
	sValue	sAttributeValue
Out	sReturnCode	Message from Teamcenter

CAA-Function:

```
HRESULT CMISupplyDrawingAttributeToTC (const CATDocument *pDocument,
                                       const CATString &sAttribute,
                                       CATString &sValue,
                                       CATString &sReturnCode);
```

Get Attribute from Teamcenter:
This Function calls *x3SupplyAttValueForObject* in Teamcenter.

In:	pDocument	CATDrawing to examine in Teamcenter
	sAttribute	sAttributeName
Out:	sValue	sAttributeValue
	sReturnCode	Message from Teamcenter

Supply File from Catia to Teamcenter

CMI provides a set of APIs which transfer a file from CATIA to Teamcenter. This functions can only be used within a customer specific CATIA application (CAA or Automation). Customizing on Teamcenter is required. (Standard Teamcenter copies the file to the Work-Location of the correspondent product/drawing)

Supply File from Catia to Teamcenter

CAA-Function:

```
HRESULT CMISupplyAdditionalFileForPrdToTC ( const CATIProduct_var spProduct,
                                            const CATString &sFullPath,
                                             const CATString &sInfoForTC,
                                            CATBoolean &bSuccess,
                                            CATString &sReturnCode);
```

This Function calls *x3GetSupplyFilePath* and *x3StoreSupplyFileCus* in Teamcenter.

In:	spProduct	CATProduct to examine in Teamcenter
	sFullPath	Full Path of file
	sInfoForTC	additional Info for TC
Out:	bSuccess	Success status of Teamcenter
	sReturnCode	Message from Teamcenter

CAA-Function:

```
HRESULT CMISupplyAdditionalFileForDrwToTC ( const CATDocument *pDocument,
                                            const CATString &sFullPath,
                                             const CATString &sInfoForTC,
                                            CATBoolean &bSuccess,
                                            CATString &sReturnCode);
```

This Funktion calls *x3GetSupplyFilePath* and *x3StoreSupplyFileCus* in Teamcenter.

In:	spProduct	CATDrawing to examine in Teamcenter
	sFullPath	Full Path of file
	sInfoForTC	additional Info for TC
Out:	bSuccess	Success status of Teamcenter
	sReturnCode	Message from Teamcenter

Post -process CATProducts for Synchronize / Update

An option is provided to perform post-processing on CATProducts that are created during Update or Synchronize. An example of this is to transfer the ownership of these files according to the parts they represent. The Post-processing step needs to be enabled in Catia by setting

CMI_PRODUCT_CREATE_POST=ON

in the CATIA environment.

When Synchronize or Update is finished, CMI will call the Teamcenter method `x3PostProcCreatePrd`. Information is provided about each CATProduct and the Teamcenter Part that it represents.

```
message x0WkBnch:x3PostProcCreatePrd(
   input :       string       className       ::
```

```
input :       string       partNumber      ::
input :       string       sPartClassName  ::
input :       string       sPartOBID       ::
input : NULL  string       sPartDbName     ::
input :       string       sPrdClassName   ::
input :       string       sPrdOBID        ::
input : NULL  string       sPrdDbName      ::
update:       SetOfStrings *vStats         ::
output:       integer      *mfail) code
```

Enable PDM-Centric Synchronize

In CMI 9.7, a new concept for Synchronize Teamcenter has been devised. The goal of this effort was to minimize Teamcenter-CATIA roundtrips. The classic Synchronize function would perform each operation individually, with the benefit of immediate user feedback in CATIA, eg. for each Part created. However, this had the downside of slow progress where complex product structures or remote sites were involved.
The new Synchronize processes the product structure entirely in Teamcenter. Only few roundtrips are necesssary, regardless of the size of the product structure.

Since the entire process has been renovated, we cannot guarantee that the new Synchronize is compatible with every existing customization, though this was naturally our goal. Therefore the new process is optional in CMI 9.7.

Setting the environment variable CMI_NEW_SYNCHRONIZE=ON in the CATIA environment will enable the new Synchronize. At the same time, it will disable the old Synchronize, and the individual Update and Create functions which are obsoleted.

Users must have message access to the PdmSessn:x3SynchFromFile message.

Validate actions before PDM- Centric Synchronize

If the Catia Configurarion variable CMI_ENABLE_VALIDATE_BEFORE_UPD=ON then the actions of synchronize are validated. The default behavior is, that all validations are successful. If an operation on an item has to be done, the kind of the operation (“CREATE” *(for creating an item)*, “UPDATE” *(for updating an item)* and “DROP” *(for delete a relation)*) is given in a parameter.

For each object one of the following methods are called:

x3ValV5Part	for Parts
x3ValV5Model	for CATParts, model, cgr, CMIArchive
x3ValV5Product	for CATProducts
x3ValV5DesTbl	for design tables

For each relation the following methods are called:

x3ValV5PartInst	for the relation between Parts
x3ValV5ModelInst	for the relation between Parts and Files (whitout CATProducts and design tables)
x3ValV5DesTblInst	for the relation between Parts or CATParts and design tables

Often used slots in the NVSET’s given in the messages above

In these methods information about the objects and related objects is provided. For details see the method’s headers in the file x3Custom.mth. If one validation fails, the process will continue until all operations are validated, but after this the program returns to the synchronize-dialog in Catia without doing any change in Teamcenter.

OBID	The obid of the object in Teamcenter. It doesn't exist, if the operation is CREATE
DB_NAME	The name of the database in Teamcenter. It doesn't exist if the operation is CREATE.
CLASSNAME	The classname in Teamcenter. If the operation is CREATE, the standard CMI-Class for the object is given.
PARTNUMBER	The partnumber in Teamcenter
EXTERNAL_LOCATION	The full path of the object. Exists, if the object is a FSItem in Teamcenter
INSTANCENAME	The instance name of a relation
NOMENCLATURE	The nomenclature of a CATProduct in Catia

Reference Geometries

The concept of reference geometries lets you include Parts and Assemblies in the CATIA product structure that are not Part of the general Teamcenter Assembly structure / Bill of Material (BOM). This allows to reference context geometry in the CATIA structure, eg. for drawing creation.

The following environment variables have to be set before startup of CATIA V5:

```
set CMI_CONFIGURABLE_NODE_BEHAVIOR=ON
set CMI_CONFIGURATION_FILE=<path>\CMICatiaV5Config.xml
```

In the CMI Configuration File (CMICatiaV5Config.xml) you need to define a Prefix for the ReferenceGeometry behavior. The following example defines a prefix of "REF_":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CMIConfigTopics SYSTEM "CMICatiaV5Config.dtd">

<CMIConfigTopics>

      <ConfigurableBehaviors>

            <ConfigurableBehavior UniqueID =
"EmbeddedNode_REF_Node">

      <BehaviorType>EmbeddedNodeBehavior</BehaviorType>
                  <PartNumberPrefix>REF_</PartNumberPrefix>
                  <Behavior>ReferenceGeometry</Behavior>
            </ConfigurableBehavior>

      </ConfigurableBehaviors>

</CMIConfigTopics>
```

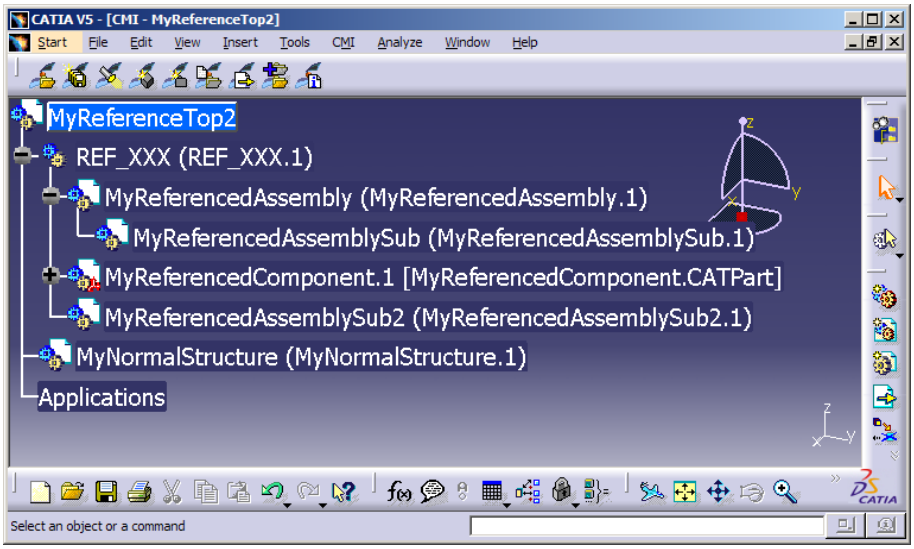

Figure 23: Reference Geometry

Parts inserted below a component prefixed with “REF_” will create a special “Has Reference Geometries” Relationship in Teamcenter (during Synchronize), instead of the standard product structure relation *g2AsmPos*. The new product structure relation is called *g2PEStrc* and is derived from *PelmStrc*. Hence it is not an *AssmStrc* relation.

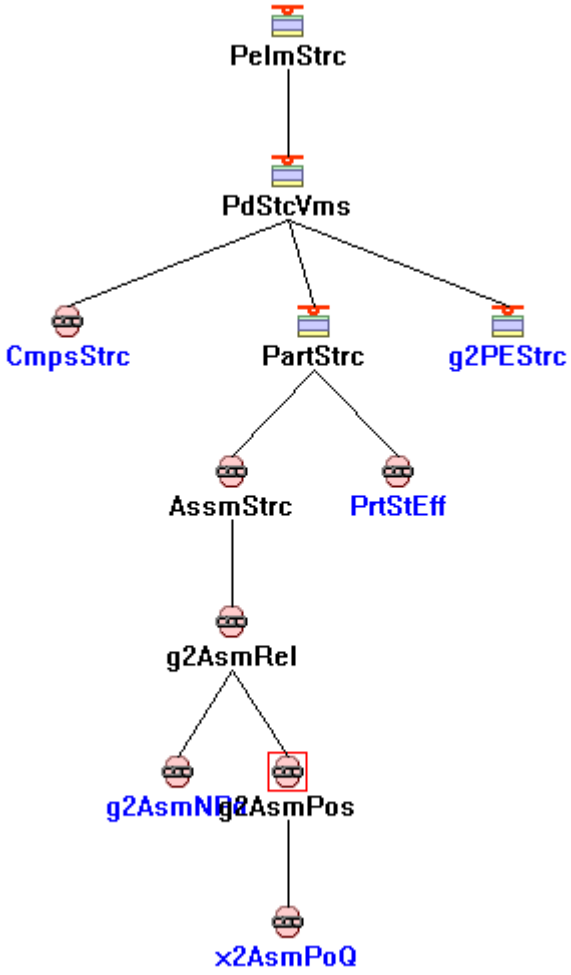

Figure 24: Class Hierarchy of PelmStrc relation

These Relations will be expanded in the CMI Workbench, but not by the *Uses Parts* expand in the standard browsers.

Observe the message access rule for the Teamcenter method *g0GenItm:g3ExpBranchMulRef* in order to remove or enable the Workbench expand relating to Reference Geometries.

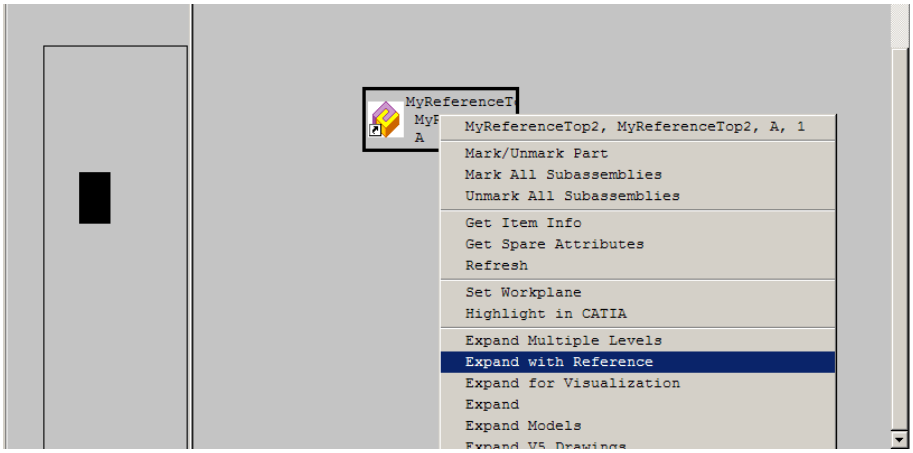

Figure 25: Expand with Reference

If you need to define your own reference geometries product structure relation, set the class constant **x0WkBnch.x4PartToRefPartC.**

Validate CATIA Version

CMI stores the CATIA Release, Service Pack and Hotfix that were used to save a CATIA file. This information can be used during Read from Teamcenter to avoid opening a file that was stored with a newer version of CATIA.

set CMI_VALIDATE_CATIA_VERSION “ON”; in the Teamcenter configuration to enable this validation. OOTB, only the Release Level is validated. You can adapt the validation scheme by overriding the following message:

```
object message g0GenBin:x3ValFileVerForRead (
      input:     ObjectPtr    CatiaFile        ::
      input:     integer      FileRelease      ::
      input:     integer      FileHotfix       ::
      input:     integer      FileServicePack  ::
      input:     integer      CurRelease       ::
      input:     integer      CurHotfix        ::
      input:     integer      CurServicePack   ::
      output:    boolean      *ReadOk          ::
      output:    string       *Message         ::
      output:    integer      *mfail
                                          )code
```

Return `ReadOk` = FALSE to prohibit loading the file. You should also return an instructive message text in `Message`, which will be displayed in CATIA.

Export to Folder

Set CMI_ENABLE_EXPORTCMD=ON to enable the *Export Structure to folder* function. The functionality allows to export CATIA data that was loaded from Teamcenter to a specific folder. Original file names can be restored thanks to the mapping file.

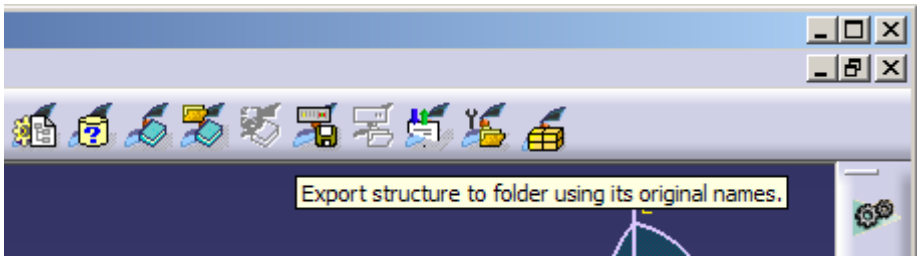

Figure 26: Export to folder

It is also possible to reset the CATIA standard attributes to their initial value before first import.
The following attributes are supported: “Nomenclature”, “Definition”, “Revision” and “Description”.
To enable the mapping of the attributes the following setting has to be added to the CATIA environment:
`SET CMI_EXPORT_STDATTRIBUTES`=<attribute1> [<attribute2> [<attribute3> [<attribute4>]]]

Name customization for Export

To enable the functionality you have to set the following environment before start up of CATIA V5:
`SET CMI_EXPORT_CUSTOMIZE_NAMING=ON`
In order to be able to use different naming schemes for the same data, an initialization method `x0WkBnch:x3GetExpNamingSchema` is called. The single purpose of this method is to optionally return an arbitrary string that distinguishes one naming scheme from another in subsequent method calls. For example, you may show a dialog where the user chooses a naming scheme.

```
class message x0WkBnch:x3GetExpNamingSchema(
   input :      string   classname          ::
   input :      string   rootPartNumber     ::
   input :      string   rootPartObid       ::
   input :      string   rootPartClass      ::
   input :      string   rootPartDbName     ::
   input : NULL string   mappingFileObid    ::
   input : NULL string   mappingFileClass   ::
   input : NULL string   mappingFileDbName  ::
   output:      string   *expNamingSchema   ::
   output:      SetOfStrings *pStats        ::
   output:      integer  *mfail);
```

with:

`classname`	x0Wkbnch
`rootPartPartNumber`	Partnumber of the top level CATProduct in CATIA
`rootPartObid`	Obid of the top level TC Part in CATIA
`rootPartClass`	ClassName of the top level TC Part in CATIA
`rootPartDbName`	DBName of the top level TC Part in CATIA
`mappingFileObid`	Obid of the mapping file
`mappingFileClass`	ClassName of the mapping file

`mappingFileDbName`	DbName of the mapping file
`expNamingSchema`	Identifier for schema to be used (can be NULL)
`pStats`	Messages displayed in CATIA

For each file in turn the following method is called, where custom values for The reference attributes can be returned. The `expNamingSchema` string from the previous method is given as an argument:

```
class message x0WkBnch:x3GetExportRefNames(
   input :      string   classname            ::
   input :      string   expNamingSchema      ::
   input :      string   tcPartNumber         ::
   input :      string   tcObid               ::
   input :      string   tcClassname          ::
   input :      string   tcDbName             ::
   input :      string   catFileName          ::
   input : NULL string   catNomenclature      ::
   input : NULL string   catRevision          ::
   input : NULL string   catDefinition        ::
   input : NULL string   catDescription       ::
   input : NULL string   extPartNumber        ::
   input : NULL string   extFileName          ::
   input : NULL string   extNomenclature      ::
   input : NULL string   extRevision          ::
   input : NULL string   extDefinition        ::
   input : NULL string   extDescription       ::
   output:      string   *partNumber          ::
   output:      string   *fileName            ::
   output:      string   *nomenclature        ::
   output:      string   *revision            ::
   output:      string   *definition          ::
   output:      string   *description         ::
   output:      SetOfStrings *pStats          ::
   output:      integer  *mfail)
```

with:

`classname`	x0Wkbnch
`expNamingSchema`	Identifier for schema to be used
`tcPartNumber`	Catia Part Number from Teamcenter
`tcObid`	TC Obid
`tcClassname`	TC Classname
`tcDbName`	TC Db name
`catFileName`	Filename in CATIA
`catNomenclature`	Nomenclature in CATIA
`catRevision`	Revision in CATIA
`catDefinition`	Definition in CATIA
`catDescription`	Description in CATIA
`extPartNumber`	External Part Number from mapping file
`extFileName`	External File ame from mapping file
`extNomenclature`	External Nomenclature from mapping file
`extRevision`	External Revision from mapping file
`extDefinition`	External Definition from mapping file
`extDescription`	External Description from mapping file
`partNumber`	CATIA Part Number to set
`fileName`	CATIA File Name to set
`nomenclature`	CATIA Nomenclature to set
`revision`	CATIA Revision to set
`definition`	CATIA Definition to set
`description`	CATIA Description to set
`pStats`	Messages displayed in CATIA

As an input, you receive the identifiers in Teamcenter (tc…); the current values of attributes in CATIA (cat…); and the original attributes from the mapping file (ext…).

The `ext…` attributes are `NULL` if no mapping file exists / is used for the exported structure. It is possible to return NULL for `partNumber, fileName, nomenclature, revision, definition, description.`

For each instance, the following method is called where you can return a new instance name:

```
class message x0WkBnch:x3GetExportInstName(
   input :      string   classname            ::
   input :      string   expNamingSchema      ::
   input :      string   relObid              ::
   input :      string   matrixIndex          ::
   input :      string   catInstanceName      ::
   input : NULL string   extInstanceName      ::
   output:      string   *instanceName        ::
   output:      SetOfStrings *pStats          ::
   output:      integer  *mfail)
```

with:

`classname`	x0Wkbnch
`expNamingSchema`	Identifier for schema to be used
`relObid`	Relation Id
`matrixIndex`	Index for Multy-Quantity
`catInstanceName`	Instance Name in CATIA
`extInstanceName`	External Instance Name from mapping file
`instanceName`	CATIA Instance Name to set
`pStats`	Messages displayed in CATIA

The new names in the exported structure are built using the following rule:

- Any CATIA attribute (Part Number, File Name, Nomenclature Revision, Definition Description) from `x3GetExportRefNames` or Instance Name from `x3GetExportInstName` that is returned (not NULL) is used in the exported structure.
- If CATIA attributes are not returned in `x3GetExportRefNames` or `x3GetExportInstName`, the values from the mapping file are used. Standard properties are only set if configured with `CMI_EXPORT_STDATTRIBUTES` and available in the mapping file.
- other the attributes currently available in CATIA are kept.

Handling of Mapping Files

Set CMI_ENABLE_CREATE_IMPORT_FILE=ON to enable the use of a name mapping file in Synchronize, Reconnect and the Export function.

To set the CATIA standard attributes Nomenclature, Revision, Description or Definition back to the original value of the initial import set the value of CMI_EXPORT_STDATTRIBUTES to the attributes you want to set back. Only used with the CMI Export Cmd.

Example: CMI_EXPORT_STDATTRIBUTES=<Nomenclature> <Definition> <Revision> <Description>

The following class constants have been introduced for the CMI Mapping File:

g0GenWb.g4MapDocClassC - Class for the Document attaching Mapping File (GenDoc)

g0GenWb.g4MapPrtDocRelClassC - Class of relation between part and Document for mapping files

x0WkBnch.g4MapFileClassC - Class for mapping files

g0GenWb.g4MapDocFileRelClassC - Class of relation between Document and mapping file

The following customizing methods have been introduced for the CMI Mapping File

Part:g3GetMappingDocs – return Documents where mapping files are attached

GenDoc:g3GetMappingFiles – return mapping files attached to Documents

x0WkBnch:x3FindMappingFile – find mapping file for a given part number

x0WkBnch:x3GetArchiveFiles - find Archives for a list of root part numbers

see x3Custom.mth for details

Syntax of the Mapping file

To open the CMI mapping file in a browser or editor with syntax check, place a file CMIMapping.dtd in the folder of the xml file.

CMIMapping.dtd:

```
<!ELEMENT mapping (references*, instances*) >
<!ELEMENT references (reference)+ >
<!ELEMENT instances (instance)+ >
<!ELEMENT reference (tcPartNumber, tcFileName, tcNomenclature,
tcRevision, tcDefinition, tcDescription,  tcClassName?, extPartNumber,
extFileName, extNomenclature, extRevision, extDefinition,
extDescription) >
<!ELEMENT instance (tcInstanceName, extInstanceName) >
<!ELEMENT tcClassName (#PCDATA) >
<!ELEMENT extInstanceName (#PCDATA) >
<!ELEMENT tcInstanceName (#PCDATA) >
<!ELEMENT tcPartNumber (#PCDATA) >
<!ELEMENT extPartNumber (#PCDATA) >
<!ELEMENT extFileName (#PCDATA) >
<!ELEMENT tcFileName (#PCDATA) >
<!ELEMENT extNomenclature (#PCDATA)>
<!ELEMENT tcNomenclature (#PCDATA)>
<!ELEMENT extRevision (#PCDATA)>
<!ELEMENT tcRevision (#PCDATA)>
<!ELEMENT extDefinition (#PCDATA)>
<!ELEMENT tcDefinition (#PCDATA)>
<!ELEMENT extDescription (#PCDATA)>
<!ELEMENT tcDescription (#PCDATA)>
<!ATTLIST reference
      id ID #REQUIRED
>
<!ATTLIST instance
      id ID #REQUIRED
      parentId IDREF #REQUIRED
      childId IDREF #REQUIRED
>
```

CATScript support

CATScript can be registered in Teamcenter and used as Part of a CMI-managed Catalog.

To insert CATScripts into a Catalog enable the *Insert CATScript from Teamcenter:*

Set CMI_ENABLE_CMICATALOGINSERTSCRIPTCMD=ON in the CATIA environment.

Two messages have been introduced for CATScript support, that can optionally be customized.

Override `x3GetScriptForInsClg` to change the way the Script is queried for insertion into the catalog:

```
class message g0GenBin:x3GetScriptForInsClg(
  input:        string       classname        ::
  input:   NULL string       sOldFilename     ::
  input:   NULL string       sOldDescription  ::
  input:   NULL string       sOldObid         ::
  input:   NULL string       sOldClassname    ::
  input:   NULL string       sOldDbName       ::
  input:   NULL string       sOldMstrObid     ::
  input:   NULL string       sOldMstrClassname::
  input:   NULL string       sOldMstrDbName   ::
  output:       ObjectPtr    *scriptObj       ::
  output:       string       *sMstrObid       ::
  output:       string       *sMstrClassname  ::
  output:       string       *sMstrDName      ::
  output:       string       *sErrorMsg       ::
  output:       integer      *mfail
      );
```

The default implementation of this method displays a Teamcenter query for Generic Script, with the supplied attributes pre-filled. Return the script object to insert into the Catalog as `scriptObj`.

Override `x3GetScriptForUseClg` to change the way the script is retrieved when the catalog is used:

```
class message g0GenBin:x3GetScriptForUseClg (
  input:        string       classname       ::
  input:        string       sObid           ::
  input:        string       sClassname      ::
  input:        string       sDbName         ::
  input: NULL   string       sMstrObid       ::
  input: NULL   string       sMstrClassname  ::
  input: NULL   string       sMstrDbName     ::
  output:       ObjectPtr    *scriptObj      ::
  output:       string       *sErrorMsg      ::
  output:       integer      *mfail
      );
```

The default implementation of this method retrieves the CATScript by OBID.

Product Bounding Boxes

With Configurable Node Behavior in CMI it is possible that CATProducts reference 3D geometry that is not stored in Teamcenter, but rather included from the CATIA environment, eg. to include Standard Parts from catalogs.

As these parts are not represented in Teamcenter, they would not be part of the result of a DMU neigborhood search. This is adressed by the CATProduct bounding boxes, which are placeholders for exactly those Parts that are external to Teamcenter.

DMU neighorhood search will expand assemblies whose products have an eligible bounding box, in addition to those assemblies that contain eligible CATParts. In particular, this is useful if you create assembly or product JT files, as it will ensure that these files are visualized when appropriate.

Configuration

In the Teamcenter configuration, set CMI_PRODUCT_BBOX “ON”; to include product bounding boxes in the DMU neighborhood search functionality.

In the CATIA environment set CMI_CALC_BBOX_FOR_IGNOREDCHILDREN=ON to enable calculation of the bounding box when a modified CATProduct is updated.

Set CMI_CALC_BBOX_FOR_IGNOREDCHILDREN=FORCE in a migration scenario for existing data. This will enable the save of bounding boxes whenever a CATProduct is writeable (to provide existing products with a bounding box, where the external parts were already present)

A bounding box will only be stored if CATParts or other geometry files are ignored due to a configurable embedded node behavior, and the new <BBox> tag is configured in the CMI configuration file (see *Configurable Behaviors in CATIA V5*)

Example:

```
<ConfigurableBehaviors>
      <ConfigurableBehavior UniqueID = "EmbeddedNode_STD_Ignore">
            <BehaviorType>EmbeddedNodeBehavior</BehaviorType>
            <PartNumberPrefix>STD_</PartNumberPrefix>
            <Behavior>IgnoreNode</Behavior>
            <BBox>true</BBox>
      </ConfigurableBehavior
</ConfigurableBehaviors>
```

In this example, CATIA component nodes with a part number beginning with STD_ are ignored by CMI. Any CATPart files below this node will not be stored in Teamcenter, but will be pulled from the environment.

During Update or Synchronize the bounding boxes of these CATParts will be combined into a single enclosing bounding box and will be stored in the CATIA Product File dataitem in Teamcenter.

Customizing

When a product is updated the bounding box can be inspected by overriding the following customizing message:

```
message x0CatPrd:x3SetCusAttrModInf (
   update: ObjectPtr  this        ::
   input : NvSet      ModelInfos  ::
   output: integer    *mfail) code
```

The ModelInfos attribute will contain the bounding box coordinates
D3D_BEX1, D3D_BEY1, D3D_BEZ1, D3D_BEX2, D3D_BEY2, D3D_BEZ2
If these values are absent from the NV Set, the product does not have a bounding box.

CHAPTER 6

Workbench Architecture

Generic Workbench

The Generic Workbench (GMI) is the base server of the CATIA Teamcenter Integration and other products of T-Systems International GmbH such as „Voxel Metaphase Integration“. The Generic Workbench contains basic classes for the following functionalities:

creating the window,

creating some dynamic items,

creating some information items about the exchange map,

creating the relations,

set/reset the communication between CATIA and Teamcenter Enterprise,

controlling of permissions,

controlling of visibility.

...

There are some public methods in the Generic Workbench which allow a customer to control the permissions and visibility. Please consult the files

custom/g3Custom.mth (GMI related methods) and

custom/x3Custom.mth (CMI related methods)

for more information.

Catia Workbench

The CATIA Workbench (CMI) is the main server of the CATIA Teamcenter Integration and bases on the Generic Workbench.

CHAPTER 7

Teamcenter Enterprise configuration variables

CMI Server Options

These are the possible settings for CMI within the PDM_CONFIG file (config.cfg):

Variable	Comment
CMILISPATH	Path to listener (cmilis will be searched in $PATH if not given)
CMICITRIX	Set to "Yes" if CMI is used with Citrix
CMILISINFO	Path to communication file for CITRIX
CMIEXMAPHOST	Host for Exchangemap with Citrix
GCVMI_SERVER_DEBUG	Set to "ON" to receive CMI debug output from the dispatcher
GCVMI_ENABLE_LINK	If set to "YES", CMI will try to create links to model files instead of copying them to the exchangemap. Recommended. Links are only available on UNIX OS.
GCVMI_MAX_QUANTITY	Maximum Quantity for CMI Multiquantity relations
CMI_CATN4D_STARTUP	Path to 4D Navigator startup script
GCVMI_SERVERPORT	Port range used on the server for CMI V4 communication
GCVMI_CLIENTPORT	Port range used on the client for CMI V4 communication
CMIDATAMODEL	DEPRECATED: Set to CMPONENTDATAMODEL to run the optional CatiaV5 centric data model (Map CATParts to Cmponent) Use CMI_BOM_PART_DEFAULT_FOR_SYNC instead.
CMI_DMU_VAULT_LOC	Vault location for DMU preprocessed file
CMI_DMU_EXP_PATH	Path of DMU_Exp utility
CMI_DMU_HOST	Host where DMU preprocessed file will be generated
CMI_DMU_STARTUP_PREF	Set to "ON" to use Startup-preferences of "super user" with DMU_Exp

CMI_CT_V5_SUBDIR	Subdirectory in Work Location for Catia V5 files (supports both Windows and UNIX format)
CMI_CT_V4_SUBDIR	Subdirectory in Work Location for Catia V4 files (supports both Windows and UNIX format)
CMI_V5_PRD_SUPPORT	Set to "OFF" to disable CATProduct management in Catia V5 (use temporary CATProducts). Doing so is not recommended.
CMI_STDPART	Set to "YES" to enable Standard Part behaviour in CatiaV5, based on StdPartIndicatorAttr in Teamcenter
CMI_CGR_HOST	Host where the vault location for CGR files resides
CMI_CGR_VAULTLOC_PATH	Path of the vault location where CGRs are searched
CMI_SHOW_BB_INCHES	Set to YES to display / enter Bounding box coordinates in inches
CMI_USE_CCS	Set to "True" for client machines or workgroups to use a CCS service for Catia V4 communication.
CMI_CREATE_DOC_INTERACTIVE	set to "ON" to create the Product Document and the CATPart Document and resp. Part objects of a BOM CATPart interactively
CMI_STORE_MULTI_INSTANCENAME	set to "ON" to store the Instance names in the x2AsmPoq Relation for the multi quantity relation (by default they are numbered (#))
CMI_USE_DB_NAME	Set to “ON” to query for objects via OBID and database name during CMI update. That means objects will be only searched within the database they are really saved in.
CMI_ITEM_INFO_ORIG	Set to "ON" to show Get Item Info of the original object (eg. Part) instead of the Catia Workbench Item
CMI_USE_BLACKBOX	Set to “ON” if you use the deprecated “BlackBox” functionality. This has a performance impact.
CMI_SINGLE_PART_DOCUMENT	Set to "ON" to reuse a single Document per Part, for multiple models
CMI_DO_NO_LINK_DRW_TO_PART	Set to "ON" to suppress the link of the Drawing to the part
CMI_USE_ORIG_NAME_FOR_PRODUCT	If set to "ON" the original filename from catia is used in Syncronize for Products and BOM Part
CMI_DESIGN_TABLES	If set to “ON”, design tables are managed by CMI. This has a performance impact.
CMI_VIEW_NETWORK_EXPAND	Set to "ON" if you require to use View Networks in the context of CAD Design. This has a performance impact during update.
CMI_TEMPLATE_VAULTLOCS	Names of vault locations for Create CATPart/CATDrawing in Teamcenter
CMI_DMU_USE_WORKLOC	Set to "ON" to create the DMU_File in the product-worklocation
GCVMI_DISABLE_CGR_LINK_V5	Set to "ON" not to Link V5-cgr-cache-files during "to Catia"
CMI_TRANSLATE_IP	If Set to 'OFF' no net_host_ipaddress_get

	(Host) is done with g3SetPortAndHost. Set if there are name resolution issues.
CMI_DMU_FILENAME_SEP	default is "_"; separator to build DMU-filename from <PartNumber>_<Revision>[_<Context>].dmu; this separator must not ocurr in PartNumber, Revision or Context
CMI_SINGLE_PART_DOCUMENT	(changed): Set to "SINGLE_FILE" to create one document for each CATPart Set to "ON" to reuse a single Document per Part, for multiple models
CMI_RESTORE_WB_WITH_ACT_CONTEXT	If set to "ON", the current configuration context is preserved while restoring a saved session. By default, the configuration context is temporarily disabled.
CMI_VIEWER_ENCODING	Encoding for the plmxml-file for the viewer. If not set, the default value is "ISO-8859-1"
CMI_ONLY_PUBLIC_TO_VIEWER	If set to "ON" the option 'Filter for public versions' in the Model Filter preference is temporarily enabled when sending to viewer.
CMI_VALIDATE_CATIA_VERSION	If set to "ON" the Catia release of the files is validated against the current sessions CATIA release. If the file is from a newer release, it is not opened.
CMI_REMOVE_GEOMETRY	If set to "DOCUMENT", if a Non-BOM CATPart is removed in CATIA, CMI will remove all Documents that attach the CATPart from the assembly , during Synchronize. If set to "MODEL", CMI will remove the Model from its Document instead. Default is OFF. *This is a default implementation of the customizing method g0GenMod:x3ProcessCATIADeletion*
CMI_PRODUCT_BBOX	If set to "ON" include Product bounding box in DMU file.

CHAPTER 8

CATIA V4 Directory Structure

Directories

The following figure shows the standard directory tree of the CATEDM installation.

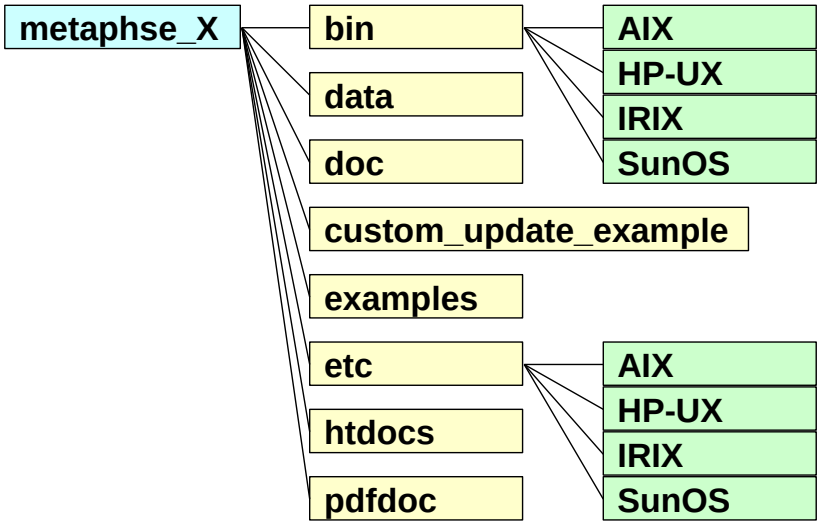

Figure 27: CATEDM installation path structure

bin/<your OS>	:	Some executables of the integration. All CATIA V4 user should have access privilege to this directory.
custom_update_example/	:	This is an example how you can edit the code and generate your own shared library.
data/	:	Declaration include files, initialization scripts, error definition file, menu definition file, message definition file.
doc/	:	The latest information about the product, script file "xt0request.sh" and a simple CATIA V4 start script “catstart.sh”.
examples/	:	Some example CATIA V4 models.

Etc/<your OS>: CMI CATIA V4 GII module in binary format (METAPHSE). All CATIA V4 user should have access privilege to this directory.

htdoc/ : Help files in HTML-Format (start page is index.htm).

pdfdoc/ : Help file in the PDF-Format (XPDF uses this file to get the context sensitive help)

Files

The Following section describes some important files and their meaning.

doc/README_FIRST.txt:

This file contains some necessary tips for the CATIA V4 integration. You should read this file.

doc/WHATSNEW.txt:

This file contains the changes of the CATIA V4 module. There are new features, changes and bugfixis.

doc/README.environment:

This file contains the possible environment settings of the CATIA V4 module. The File **README.env.xls** is the same in the Micosoft Excel format.

data/xt0request.sh:

This file is needed for the communication between CATIA V4 client and Teamcenter Enterprise server. You should modify this file and copy it into any directory in search path of CATIA V4 clients.

data/ini.env:

This initialization script contains some necessary environment settings for the integration. The meaning of the certain environment variables is described in the **ini.env** file. This file should be adjusted and each user must run this script before starting CATIA V4. Please refer the **README.environment** file for more information.

data/METAPHSE.include:

CATIA V4 declaration file for the integration. This file should be included to the users **USRENV.dcls** or to any other local or global CATIA V4 declaration file.

data/appdefault.obj :

This file contains some environment settings of the CATIA V4 module. Please refer the **README.environment** file for more information. The used syntax is "**{<parameter>} <value>**". Some parameters are named **<parameter>.example**. If you want to set this parameter you have to delete the extension ".example".

data/ERREDB :

This file contains the error messages in CATIA V4.

data/dshdrawingframe.sh :

This file contains some information how to fill a drawing title block.

data/cleanbox :

All text inside the boxes described in this file will be deleted during filling the drawing frame.

data/plotconf:

This file contains the options of the CATIA V4 PLOT UTILITY.

data/catiaedb.msg :

This file contains a list of all used messages appearing in CATIA V4 when you are using the CMI Module. You can customize this message file to your own needs.

data/catiaedb.menu :

This file contains a list of all menu points of the CMI Module. You can change each menu point to your own needs. A menu name consists of 8 characters at maximum. An empty entry means that the menu point is disabled.

data/edmhelp.conf :

If xpdf is used as help tool, this file contains the bindings between the menu and the pages of the help file. If you write an own help file you must edit this file.

$HOME/exchangemap :

This directory is a local UNIX directory in user's home directory. On each CATIA V4 client workstation an exchange map must exist. The task of this UNIX directory is to exchange data between CATIA V4 and Teamcenter Enterprise. Each user should have an own exchange map.

$HOME/.dshcatiarc.obj :

This optional script file overwrites the default settings for the user. Please refer the **README.environment** file for more information.

Modify CATIA V4 Environment

extend STEPLIB and CATDEC environment settings as following (example):

```
STEPLIB= ..... :/usr/lpp/catia/v4r1_code/gii/steplib
STEPLIB=$STEPLIB:/catia/gii/metaphse_catedm/etc/<your OS>
export STEPLIB
CATDEC=$CATDEC: /catia/gii/metaphse_catedm/data
export CATDEC
```

Include the file **data/METAPHSE.include** to each user's **USRENV.dcls** file as following (example):

```
/*-------------------------------------------------*/
/* USRENV.dcls     DECLARATION FILE                */
/*-------------------------------------------------*/
INCLUDE ('/catia/v4r1/prod/USRENV.include');

/*-------------------------------------------------*/
/*  INCLUDE all other Configuration – files        */
/*-------------------------------------------------*/
INCLUDE ('/catia/gii/metaphse_catedm/data/METAPHSE.include');
/*-------------------------------------------------*/
```

User dependent configurations

Normally you don't need any modifications for a certain user (except user's personal **USRENV.dcls** file). Therefore, each user can overwrite the default settings. You may copy the file **data/appdefault.obj** to the user's home directory and rename it to **.dshcatiarc.obj**. Now you can edit this file and overwrite the existing settings.

Following the order of running the setting files:

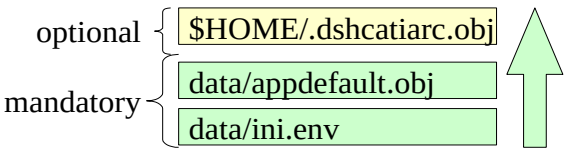

Figure 28: Initialization files with their order.

The following Environment settings are possible (see next tables)

Name of Environment Variable (ini.env)	Name of **CONFIGURATION FLAG (APPDEFAULT.OBJ .DSHCATIARC.OBJ)**	Short Description	Possible Values
CATEDM_DEBUG	Debug	set a debug level	0 - no debug 66 - full debug
CATEDM_BINDIR	-	path to the executables of CATEDM installation	
CATEDM_SCRIPTDIR	-	path to the scripts and configuration files of CATEDM installation	
CATEDM_CONN	Connectmethod	method to connect with CMI workbench	

CATEDM_EXCHANGEMAP	Exchangemap	directory were CMI and CATIA V4 exchange model files	
CATEDM_EXMAPDLNAME	exchange_dlname	logical map name which is realated to the exchange map	
CATEDM_MLINKS	Modellinks	create a symbolic link (description) to each model in exchange map	
CATEDM_TMPAXIS	Tmpaxis	use a start model	0 - no start model 1 - use a start model
CATEDM_TMPAXISDLNAME	tmpaxis-dlname	logical CATIA V4 map for the TEMPAXIS model	If there I no TMPAXIS MODEL (MAP) defined the INITIAL_MODEL of CATIA V4 is used
CATEDM_TMPAXISDSNAME	tmpaxis-dsname	CATIA V4 map for the TEMPAXIS model	
CATEDM_TMPAXISNAME	tmpaxis-name	template for the TMPAXIS model: the CATIA V4 model without ".model" extension	
CATEDM_NAMETYP	modelnametype	position from which a CATIA V4 model name is displayed in CATIA V4 status	
		field. usefull for names longer than 32 types.	
CATEDM_LOGF	Logfile	name of the file were the log should go.	
CATEDM_HELPERTOOL	Helper	name of the installed helper application (xpdf, netscape, acroread)	
CATEDM_HELPFILE	Alias	name of the online help file	
CATEDM_CATIAVER	Catiaversion	to set the current CATIA V4 version	
CATEDM_PROJECT	Project	defines the name of the current CATIA V4 project	
CATEDM_WPMODUS	wpmodus	default workplane modus	0 - assembly 1 - default workplane 2 - geometry 3 - multiselection
CATEDM_GEOPOS	geoposallowed	geometry position support	0 - off 1 - on
CATEDM_APPDEF	-	name of the CATEDM configuration file (default is appdefault.obj)	
CATEDM_RCFILE	-	path and name of the user specific configuration file	
		(default is $HOME/.dshcatiarc.obj)	
CATEDM_CONNECTXFILE	connectx-script	alternative way to connect to the workbench – not used yet	
CATEDM_NORMPART	normpart-support	support for norm part integration NIS	0 - off 1 - on
CATEDM_REFRESH	autorefresh	refresh the current CATIA V4 session after a CATEDM READ action	0 - off 1 - on

CATEDM_RM_MODELS	remove-models	remove models from exchange map at first CATEDM startup time	0 - off 1 - on
CATEDM_ASSEMBLY_ SYMMETRY	assembly-symmetry	allow MOD POS->MOVE- >SYMMETRY for assembly positions	0 - off 1 - on
CATEDM_LOADWARNING	load-warning	show a warning panel when a read action is started from workbench	0 - no warning 1 - at READ 2 - at REREAD 3 - at READ and REREAD
CATEDM_MERGE	merge-support	CATEDM supports CATIA V4 MERGE	0 - off 1 - on
DebisLICDIR	-	location of debis licman license file	
DebisLICBIN	-	location of debis licman executables	
CATEDM_LICMAN_START	license-startscript	license manager start script - default is licman12	Licman12
CATEDM_LLD_AUTOSTART	lld-autostart	start local license daemon at first CATEDM startup time	
CATEDM_SETUPSTAT	setupstatus	reads configuration flags from appdefault.obj at any module entry only used for debug	0 – off 1 - on
CATEDM_DMPF	stdumpfile	information file - only needed for VMI	
CATEDM_EDBHOST	edbapphost	CATIA V4 client host name	
CATEDM_SMD_WEIGHT	smd-weight-support	CATEDM supports to read the weight and the position of SMARAGD models	0 – off 1 - on
CATEDM_DESCINFOAPP	desc-info-applications	read the user defined blocks and write the data to the info object. To accsess to the blocks you need three things: 1. The application string: before any description, modification or read rotine is used, the user must declare the application string 2. Element type 3. Types of the description: (1-16000) This routine restricts the size of the data block to 32 elements of each type.	application list
CATEDM_DESCINFOELE	desc-info-elements		pt, ln, ...
CATEDM_DESCINFODESC	desc-info-descriptions		types of the descriptions
	example:	CATEDM_DESCINFOAPPEDBCATIA CATIAEDB CATEDM_DESCINFOELEPT LN TXTN CATEDM_DESCINFODESC{12345 2456} 3457 {1111} "==> PT: search for application string EDBCATIA and CATIAEDB and types 12345 and 2456	

CATEDM_COMMENT	comment-support	Read the comment lines of a model file and write them into a file into the exchangemap directory. The full filename stands into the info obj	0 – off 1 - on
CATEDM_NEWUPD	Newupd	After your selecting METAPHSE > UPDATE > ALL a list of models that have been modified will come up in CATIA V4. If some models were active, but the user mustn't change these models, a status panel will pop up. It is not necessary to confirm this panel. After your confirmation the selected models will be updated in Teamcenter.	1 - in CATIA V4 (default) 0 -no selection of the models in CATIA V4
CATEDM_REPLACEMODEL	replace-model	Save As / Create can replace the original CATIA V4 model in the current SESSION by the new registered Teamcenter CATIA V4 model. (This is only possible if the model was loaded by CATIA FILE->open) If the replace functionality is turened off: The new model is loaded additional into the CATIA SESSION	1 - replace (default) 0 - do not replace
CATEDM_CUSUPD	cmi-custom-update	calls the shared lib libcmi_custom_update.<a,sl,so> The customer can use its own libs to perform some CATIA action at update / create	1 - use the shared lib 0 - do not use the shared lib
CATEDM_BBOX	bbox-support	bounding box generation for CATIA models (2 points for each model: D3D_{X,Y,Z}{1,2}) bounding box points will be sent for each model at UPDATE ALL/MODELS, CREATE/SAVE AS and MULT CRE.	0 - no bbox support at all 1 - standard bbox support (without checkbox at update and muticreate) 2 - bbox support with checkbox (default support off) 3 - bbox support with checkbox (default support on)

CATEDM_BBOXLAYER	bbox-layer-list	Only the elements of these layers are used to generate the bounding box. If no layer is set the current layer is used.	0 1 2 ... 254 - creates a bounding box using these layers -1 - uses the actual layer filter -2 - creates the box using all layers

This is a short documentation of the configuration environment of CUSTOM ATTRIBUTS		
ATTRIBUTE	VALUE	DESCRIPTION
VolumeMass	Type: REAL CATIA V4 uses the units of the specific model	1. Sets the model standards 2. Calculates the weight and the Center of Gravity (COG) of the Volume (VOL) and the Polyhedral and exact solid (SOL) The results are written into the info object: {SOLWEIGHT} value {SOLCOG} {x-value} {y-value} {z-value} {VOLWEIGHT} value {VOLCOG} {x-value} {y-value} {z-value}
SurfaceMass	Type: REAL CATIA V4 uses the units of the specific model	1. Sets the model standards 2. Calculates the weight and the Center of Gravity (COG) of the Surface (SUR), the Face (FAC), the Skin (SKI) and the SPACE polyhedral surface (POL) The results are written into the info object: {SURWEIGHT} value {SURCOG} {x-value} {y-value} {z-value} {FACWEIGHT} value {FACCOG} {x-value} {y-value} {z-value} {SKIWEIGHT} value {SKICOG} {x-value} {y-value} {z-value} {POLWEIGHT} value {POLCOG} {x-value} {y-value} {z-value}
cmi-custom-update	Type: INTEGER	if [VAL] > 0 then call the functions of the libcmi_custom_update.<a,sl,so>. If the configuration flag {cmi-custom-update} is set to 1 or 2 this attribute will have no effect.
CMI_DRAWING	Type: INTEGER	possible values: 1 --> the drawingframe of this model will be updated 0 --> the drawingframe of this model will be unchanged -1 or no attribute --> the behavior belongs to the environment

User dependent API in CATIA

There are different ways to call the CATEDM custom functionality:

- Edit the file metaphse_4.x.x/data/appdefault.obj

{cmi-custom-update} 1	Calls the functions
	void cmi_custom_update(int *ERROR);
	at UPDATE ALL / UPDATE GEOMETRY / UPDATE ACTIVE
	and void cmi_custom_create(int *ERROR);
	at DOCUMENT CREATE / DOCUMENT SAVE_AS / DOCUMENT MUL_CRE

{cmi-custom-update} 2 (catedm version 4.0.3 and later)	Calls the function void cmi_custom_update(int *ERROR); at UPDATE ALL / UPDATE GEOMETRY / UPDATE ACTIVE
{cmi-custom-update} 3 (catedm version 4.0.3 and later)	Calls the function void cmi_custom_create(int *ERROR); at DOCUMENT CREATE / DOCUMENT SAVE_AS / DOCUMENT MUL_CRE

- overwrite the message **x0CTFile:x3GetCustomDataForCAD** in the x3Custom.mth file of the CMI installation. (catedm version 4.0.3 and later) It is possible to sent the FLAGS

```
attributelist = [{attr_1} [{attr_n}]] {cmi-custom-update}
valuelist     = [{value_1} [{value_n}]] {1}
```

  with READ / REREAD to CATIA (needs customisation in MP)

  At UPDATE ALL / UPDATE GEOMETRY / UPDATE ACTIVE of the related model the function "**void cmi_custom_update(int *ERROR);**" is called.

CHAPTER 9

CATIA V5 Directory Structure

Directories

Following figure shows the standard directory tree of the CMI CATIA V5 (CMICATV5).

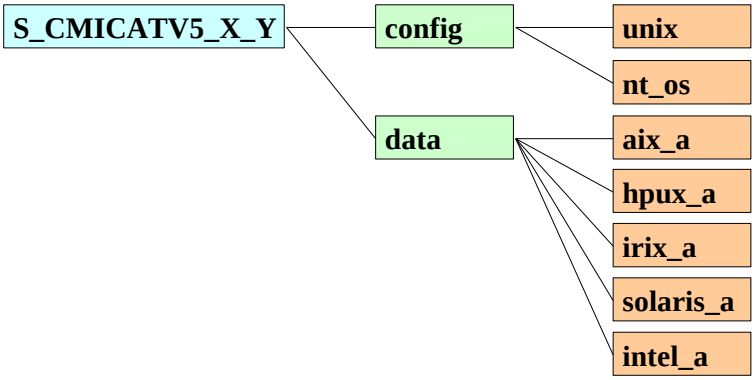

Figure 29: Directory structure of the CMICATV5 module

The **config** directory contains sample CATIA V5 Configuration files for unix and Windows 2000. The **unix** configuration contains two sample environment settings for the shells **sh** and **csh**. The **nt_os** configuration contains a sample CATIA V5 Environment file for the CATIA V5 Environment editor.

The data directory contains the binary distributions for the CMICATV5 module for the supported operating system mnemonics.

The supported operation systems and their mnemonics are:

AIX5.1	aix_a
HPUX 11	hpux_b
IRIX 6.5	irix_a
Solaris 7	solaris_a
Windows 7 / XP	intel_a / win_b64

The mnemonic “aix_a” will be chosen as an example of a CMICATV5 installation directory on Unix.

Following figure shows the directory tree of ”aix_a”.

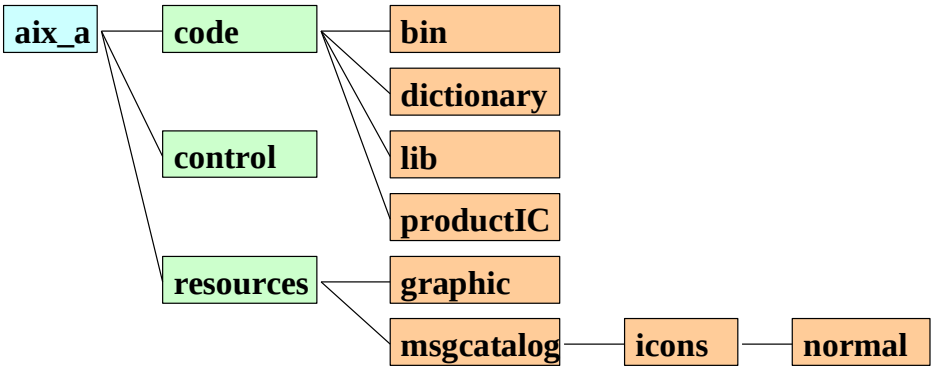

Figure 30: Directory structure of the CMICATV5 installation directory

`msgcatalog/` : If you want to change Text Messages in CATIA V5 the files with the extension CATNls in the `msgcatalog` directory can be changed. This message files contain string variables. After adapting this strings changes will be shown in CATIA V5 (see examples below!).

All other files in the CMICATV5 installation directory should not be touched.

Files

This section describes some important files and their meaning. The Files have a Text Message catalog with messages displayed in CATIA V5.

The following example shows the content of the file CMIUpdateCommandHeader.CATNls:

```
// (c) T-Systems 2001


//====================================================================
// Message catalog for the CMIAddinHeader command headers of the
// CMIAddin addin
//====================================================================


CMIUpdateCommandHeader.CMIUpdateCommandHeader.Category = "CMI";
CMIUpdateCommandHeader.CMIUpdateCommandHeader.Title    = "Update";
CMIUpdateCommandHeader.CMIUpdateCommandHeader.ShortHelp= "Update Teamcenter";
CMIUpdateCommandHeader.CMIUpdateCommandHeader.Help     = "Update Teamcenter";
CMIUpdateCommandHeader.CMIUpdateCommandHeader.LongHelp = "Update
This command updates positions and files in Teamcenter.";
```

If the mouse pointer is over the tool icon (in this example: the **"Update Teamcenter"** icon) the **Title** you will see in the status line before the command line. The **ShortHelp** messages will appears in the tooltip and the **Help** message appears in the status line left.

After moving the "What's This?" icon to the toolbar icon the text in **Long Help** will be shown.

The following changeable files have the same structure as this example file.

`resources/msgcatalog/CMIReadCommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"Read from Workbench" .

`resources/msgcatalog/CMIUpdateCommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"Update Teamcenter" .

`resources/msgcatalog/CMIUpdateCreateInteractiveCommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"Sychronize Teamcenter" .

`resources/msgcatalog/CMICreateCommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"Create in Teamcenter" .

`resources/msgcatalog/CMISaveAsCommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"SaveAs in Teamcenter" .

`resources/msgcatalog/CMICreateV4CommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"Create V4 Model" .

`resources/msgcatalog/CMISaveAsV4CommandHeader.CATNls` :

This file contains the Text Messages catalog for the CMI Command

"SaveAs V4 in Teamcenter" .

resources/msgcatalog/CMIInfoCommandHeader.CATNls :

This file contains the Text Messages catalog for the CMI Command

"CMI-Info" .

For working with Text Messages in different languages the files in the directory **resources/msgcatalog** should be copied in different subdirectories and adapted there.

Following figure shows an example of possible subdirectories under **msgcatalog.**

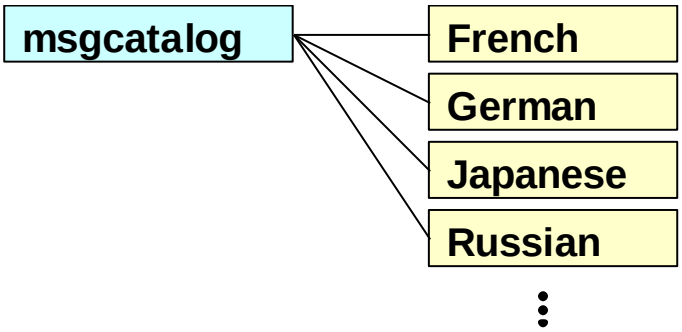

Figure 31: Example of directory structure of the CMICATV5 installation subdirectory msgcatalog

The English message files are located directly under the directory **msgcatalog.**

Customer dependent configurations for CATIA V5

Environment settings

Environment Variable	Comment
CMI_CREATE_V4_WITH_PARENT	If set the CreateV4 Command only works in the Product Structure with a selected CatPart
CMIXMAP	The location of the Catia V5 Exchange Map
CMI_DEBUG	If Set to 'ON' the Debug output is written to stdout, else no output is created
CMI_REMOVE_CMIREADCMD	If Set the Read Command is not available
CMI_REMOVE_CMIUPDATECMD	If Set the Update Command is not available
CMI_REMOVE_CMICREATECMD	If Set the Create Command is not available
CMI_REMOVE_CMISAVEASCMD	If Set the SaveAs Command is not available
CMI_REMOVE_CMIUPDATECREATECMD	If Set the Synchronize Command is not available
CMI_REMOVE_CMICREATEV4CMD	If Set the CreateV4 Command is not available
CMI_REMOVE_CMISAVEASV4CMD	If Set the SaveAsV4 Command is not available
CMI_REMOVE_CMIINFOCMD	If set the Info command is not available

CMI_REMOVE_CMIHLINWBCMD	If set, the Highlight in Workbench command is not available
CMI_REMOVE _CMIUPDATECREATEINTERACTIVECMD	If set, the Update and Create Interactive command is not available
CMI_REMOVE_CMIINFOCMD	If set, the CMI Info command is not available
CMI_SAVEAS_V4_WITH_DIRTYCHECK	If Set the SaveAsV4 Command only works with saved CatParts
CMI_SAVEAS_V4_WITH_CMICHECK	If Set the SaveAsV4 Command only works with CatParts from CMI
CMI_USE_COMMAND_SUBMENU	if Set to "OFF", button similar-command grouping is turned off in the toolbar.
CMI_CALC_BBOX	Calculate and save bounding box info at CATParts on update (for DMU)
CMI_DISABLE	Set to 'ON' to get NO CMI toolbar. No check for CMI license is done.
CMI_ENABLE_VP_SAVE	Set to '1' to get the Functionality Save Virtual product
CMI_CALC_SHEET	When a drawing is saved, make the list of sheets available in Teamcenter
CMI_REFERENCE_OF_DRAWING	When a drawing is saved, make the referenced Products/Parts available in Teamcenter
CMI_BOUNDING_BOX_EXCLUDE_HIDDEN	Set to "ON" to exclude hidden objects from the bounding box calculation
CMI_DISABLE_HIDESHOW	Set to "ON" to load the hide /show status as in the product (products not in the CMI Workbench are not hidden)
CMI_DISABLE_SET_TIMESTAMP	Set to "ON" to disable to set the last modification date to the files in the exchangemap.
CMI_REMOVE_CMIADDTEMPCMD	Set to "ON" to disable the AddTemp Command
CMI_ADDTEMP_PRAEFIX1	Default is "TMP" the first praefix for the rename of the Partnumbers and Filename for the AddTemp command
CMI_ADDTEMP_PRAEFIX2	Default is "_" the second praefix for the rename of the Partnumbers and Filename for the AddTemp command The Default Praefix is set to TMP#_ # is a counter in catia
CMI_ENABLE_CHECKMODTIMESTAMP	Default is "ON". Set to "OFF" to disable the fuctionality to check for "Saved" Files, which are saved by the native CATIA Save command; also, if Set to "OFF" out of sync cgr files in the local cache are not treated by CMI.
CMI_ENABLE_CACHEMODE_RESETDOCLINKS	Default is "OFF". Set to "ON" if it should be attempted to reset(refresh) document links in Cache Mode. Default is OFF as R12 sp3 can't refresh geometric document links properly
CMI_ENABLE_CMIUPDATEPOSITIONCMD	Default is NULL. Set in "ON" to enable the Update Position Command
CMI_ENABLE_CMICREATEFORDOCCMD	Default is NULL. Set to "ON" to enable the Create For Doc Command
CMI_ENABLE_CMIGETORIGGEOCMD	Default is NULL. Set to "ON" to enable the Get orig. Geometry Command

CMI_USERELEASEDCACHE	Default is NULL. Set to "ON" to transfer CGR-files to Released Cache
CMI_RELEASEDCACHEDIR	Only used if CMI_USERELEASEDCACHE=ON.Sets the Released Cache dir used by CMI to a specific member of the list of Released Cache directories in CATIA. Default: not set -> CMI uses the first member of the list. If set to a member, this member must be part of the list.
CMI_CLEANRELEASEDCACHE	Only used if CMI_USERELEASEDCACHE=ON.Default is "OFF" Set to "ON" if you do not use version independent filenames
CMI_CREATETEMPCGRCOMP	Only used if CMI_USERELEASEDCACHE=ON.Default is "OFF" Set to "ON" to use temporary CATIA components which contain the related CGR as shape representation, instead of the original CATPart.
CMI_CHECK_LINKED_DRAWING	If set to "ON" then with Create / Create & Link / Save As the CatPart is checked for a related opened Drawing
CMI_CHECK_LINKED_PRODUCT	If set to "ON" then with Create / Create & Link / Save As the CatPart is checked if it is opened in another product
CMI_DEFAULT_UNIT	If no unit of measurement is passed from the PDM system to CATIA, then the unit may be set here.
CMI_CONFIGURATION_FILE	Full filename and path to the CMI XML Configuration file (alternative/complement to system environment variable declaration)
CMI_ENABLE_SINGLEPARTMODUS_READ	If set to "ON" , the "Single Part Modus" option in the CMI Options CATIA V5 property page is enabled
CMI_CONNECTPDM	String to use to override the default command-line omfcl call
CMI_ENABLE_NATIVEPRODUCTTRAFO	Set "ON" to enable the option to suppress CMI Transformation
CMI_ENABLE_APPLY_VISUMODE	If set to "ON", CMI will switch CATParts that it loads in Design mode, back to Visualization mode if possible (supported by CATIA beginning R13)
CMI_RESTORE_POSITION	"ON" / "OFF" --> if "ON" user can reset Matrix position to original CMI Matrix position within CATIA V5
CMI_ENABLE_REMOVED_DOCCOM	Default is "ON", Set to "OFF" will disable support for removed Models in TC, if "ON" the removed models will be highlighted at Read Command
CMI_DRAWING_CHECKUPDATEOFASSEMBLY	If Set to "ON" CMI will warn if you update a CATDrawing when at least one related CATPart / CATProduct is not saved yet
CMI_ENABLE_UPDATEPOSITIONDIALOG	If set to "ON" then show a dialog of modified positions at update
CMI_DISABLE_LOAD_STDCATPARTS	If set to "ON" then standard CATParts will not be loaded into design mode automatically. They are identified not by their part number (which is not available in cache mode) but by their instance name, which must be "Part Number + .(dot) + some string".
CMI_IGNORE_NONCMI_ROOT_CHILDREN	If set to "ON" and "Use Virtual Root" is turned on in the CMI Settings in CATIA, then any children attached to the virtual root that are not from CMI will be ignored during Update/Sync commands
CMI_DISABLE_SAVETOXMAP	Set to "ON" if files located outside of the exchange-map should NOT be moved into the exchange map before they are Created/updated in the PDM System

CMI_PACK_ARCHIVE_CMD	String to use to override the default command-line for packing/zipping archive files. Default is "zip -0 -q -j", where -0 is "store only", -q is "quiet operation", -j is " junk (don't record) directory names"
CMI_UNPACK_ARCHIVE_CMD	String to use to override the default command-line for unpacking/unzipping archive files. Default is "unzip -o -j -q -d", where -o is "Override without prompting", -j is "do not use Directory names", -q is "quiet mode", -d <Dir> "extract to dir"
CMI_REMOVE_CMIATTARCCMD	If set, the Attach Archive command is unavailable
CMI_GLOBAL_DISABLE	If set to "ON", the CMI General Update Addin commands are disabled
CMI_DISABLE_REPLACE_WRONG_PRD	If set to "ON", the following functionality is disabled: During a Read, if CMI recognizes that a file with a different UUID has been received from the PDM System instead of the file UUID named in the parent CATProduct, then the new file is attached in place of the old.
CMI_ENABLE_VALIDATE_BEFORE_UPD	Set to "ON" to enable customer specific validation of all operations in Synchronize Command before a sync may be executed.
CMI_DISABLE_STEP_SYNC	Set to "ON" to remove the "Synchronize operations singly" check box from the Synchronize dialog
CMI_DISABLE_NEW_CGR	If set to "ON", this disallows the addition of new CGR files to the product structure
CMI_DISABLE_NEW_CGR_INSTANCE	If set to "ON", this disallows the addition of new instances of CGR files to the product structure
CMI_DISABLE_NEW_V4MODEL	If set to "ON", this disallows the addition of new V4 Model files to the product structure
CMI_DISABLE_NEW_V4MODEL_INSTANCE	If set to "ON", this disallows the addition of new instances of V4 Model files to the product structure
CMI_DISABLE_ANALYSIS_IN_ARCHIVE	If set to "ON" the support for CATAnalysis in CMIArchives is disabled.
CMI_ENABLE_ANALYSIS_COMPUTATION_IGNORE	If set to "ON" the the computations in CATAnalysis is ignored, else the computations must be deleted by the user.
CMI_DISABLE_MODEL_IN_ARCHIVE	If set to "ON" the support for V4 models in CMIArchives is disabled.
CMI_DISABLE_CGR_IN_ARCHIVE	If set to "ON" the support for cgr in CMIArchives is disabled.
CMI_ENABLE_ARCHIVE_ROOT_PRODUCT_ONLY	If set to "ON" the Root in CMIArchives must be a CATProduct.
CMI_REMOVE_CMIMODNCARCCMD	If set to "ON" the Modify non CATIA command is not available
CMI_ENABLE_UPD_MODELSELECT_DIALOG	If set to "ON" a dialog for model update selection is shown in update command
CMI_ENABLE_CMIMULTIQTYCMD	If set to "ON", the Build as Multi Quantity command is available
CMI_ENABLE_CHECKISUPTODATE	If set to "ON" the user can cancel the Update /Create / Save As action if a CATPart / CATProduct is not synchronized in the CATIA Session
CMI_ENABLE_CMIEXTERNALDOCCMD	If set to "ON" the command to load referenced documents is available. Additional software is required for this.
CMI_DISABLE_LOADOK_MESSAGE	If set to "ON", no message is shown after a successful Read from Teamcenter
CMI_DISABLE_UPDATE_WB	If set to "ON", the Workbench is not updated with new items created during Synchronize
CMI_ENABLE_CHECKISUPTODATE	If set to "ON", a check is performed during Update if any geometry needs to be updated in Catia

CMI_ENABLE_CHECKMULTIEMBARC	If set to "ON", ambiguous (same partnumber) local components are disallowed in an archive
CMI_GETPOINTEDDOCUMENTS	if set to "ON", dependencies based on referenced documents are created in Teamcenter. This requires additional software.
CMI_ENABLE_CHECK_PRD_VAL_IGNORE	If set to "ON" the validate function of Synchronize succeeds if a product with no structural changes is modified and has to be updated
CMI_ENABLE_CMIOPTIONSCMD	If set to "ON" the CMI Options dialog command button is available
CMI_ENABLE_STDPARTINFO	If set to "ON" the standard Part infos are requested from TC, needed for the Component standard part integration
CMI_REMOVE_CMIUSEPDMSTRUCTURECMD	If set to "ON" the "Insert from Teamcenter"-Command is not available
CMI_ENABLE_RESETINVALIDPOSCMD	If set to "ON" the "Reset Invalid Position"-Command is available
CMI_ENABLE_CHECK_INVALID_POS	If set to "ON" during Update all models are checked for invalid positions
CMI_BOM_PART_DEFAULT_FOR_SYNC	Set the default value for new Models in the sync dialog: NOT_SET, BOM, all ather settings are normal models
CMI_ENABLE_ASK_FOR_BOM_PART	If set to "ON" the user is asked which kind of Part should be created in TC
CMI_ENABLE_CMIARCHIVE_CREATE	If set to "ON" it is possible to create (Attach) archives without parent and without CMI Parent
CMI_CAT_ENV_SCRIPT	points to the cat start file (full path with extension) which is used to start the CMISender executable(Used in Omf and CNEXT environment), CNEXT startup will create the file for the actual CATIA and the Omf will use this for starting CMISender
CMI_DISABLE_CMIBUILDVISUCMD	If set to "ON" the CMI BuildVisu Command is not available
CMI_DISABLE_CREATE_CATDRAWING	If set to "ON" the creation of new CATDrawings is disabled
CMI_DISABLE_CREATE_CATPART	If set to "ON" the creation of new CATParts without Parents is disabled
CMI_DISABLE_CMIRECONNECTCMD	If set to "ON" the CMIReconnect Command is not Available
CMI_ENABLE_ARCHIVE_CACHE	If set to "ON" the archive file can handle Release cache files
CMI_PDM_MANAGED_STDCATPARTS	If set to "ON" then CMI treats standard part geometry (CATPart files) like regular component part geometry, that is, the geometry is expected to be attached to a document describing the Component in PDM and it is transferred to the exchange map at "To Catia".
CMI_PACK_ADD_ARCHIVE_CMD	String to use to override the default command-line for Adding files to archives. Default is "cmi_zip -0 -q -j", where -0 is "store only", -q is "quiet operation", -j is " junk (don't record) directory names"
CMI_PACK_ARCHIVE_MAX_FILES	Define the maximal number of files which will be packed with the Zip command. If not set, Default is 100. Used to reduce the needed length of the command line for the system call.
CMI_NO_PUBLIC_POS_UPDATE	Do not try to update transformations under products that are read only
CMI_PACK_ARCHIVE_MAX_COMMANDLINE	Define the maximum length (in Bytes) of the system call which will pack a CMI Archive. If not set use system limits of the OS. If the limit is reached, the pack Archive Command will be split (see CMI_PACK_ADD_ARCHIVE_CMD)
CMI_ARCHIVE_AUTOUPDATE	Automatic update after "Attach an Archive"

CMI_REPLACE_WRONG_PRD_AUTO	If set to "ON" The confirmation dialog for the following functionality is disabled: During a Read, if CMI recognizes that a file with a different UUID has been received from the PDM System instead of the file UUID named in the parent CATProduct, then the new file is attached in place of the old
CMI_ENABLE_DEACTIVATED_CHECK	If set to "ON" the active window is searched for deactivated products, if deactivated are found, the update operation is not allowed
CMI_ENABLE_PARTINFO_FOR_MODEL	If set to "ON" the more info button will provide information about the Part in TC (instead of only the data item)
CMI_DISABLE_DT_IN_ARCHIVE	If set to "ON" CMI does not store DesignTables in Archive even if CMI_DESIGN_TABLES is set in Teamcenter
CMI_USE_DTFORPRODUCT	If set to "ON" CMI also handles DesignTables for CATProducts if CMI_DESIGN_TABLES is set in Teamcenter
CMI_REP_FORMATS	Sets the Representation formats, which should be handled. Example: "{wrl} {stl}"
CMI_DISABLE_REP_IN_ARCHIVE	If set to "ON" the support for Representations in CMIArchives is disabled.
CMI_REMOVE_UNKNOWN_FILES_IN_XMAP	If set to "ON" unknown files in CMIXMAP will be subject to cache size management
CMI_XMAP_CACHE_SIZE	Maximum size in MB of cached files in CMIXMAP after closing CATIA
CMI_RELMAP_CACHE_SIZE	Maximum size in MB of cached cgr files in CMI_RELEASEDCACHEDIR
CMI_DISABLE_CMISAVELOCALCMD	If set To "ON" the CMI SaveLocal Command is not available
CMI_DISABLE_CMIRESTORELOCALCMD	If set To "ON" the CMI RestoreLocal Command is not available
CMI_USE_91_TOOLBAR	If set to "ON" the 9.1 Toolbar style (only one toolbar) is used
CMI_GET_BOMTYPE_FROM_TC	Set to "ON" to get the Bom-Type from Teamcenter for new Catia-Files
CMI_ENABLE_CMICATALOGREADCMD	Set to ON will enable the Catalog Read Command in CATIA V5.
CMI_ENABLE_CMICATALOGUPDCRECMD	Set to ON will enable the Catalog Create/Update Command in CATIA V5
CMI_DELETE_STALE_DESIGNTABLES	If set to "ON" DesignTables in CMIXMAP that are no longer referenced by a CATPart/CATProduct will be deleted.
CMI_USE_FILENAME_WINDOWTITLE	If set to "ON" the title of windows loaded by CMI shows "CMI - <Filename>" instead of the default: "CMI - PartNumber>"
CMI_ENABLE_CMICATDUAREAD	If set to "ON" Enable the CMICatDuaRead Command in the toolbar
CMI_CATDUAV5_CONFIG	Full path to the CATDUAV5 configuration file
CMI_CATDUAV5_RESULTVIEWER	Viewer to show the CATDUA result (Windows optional / unix required). If set to "OFF" no result is shown
CMI_CATDUAV5_COMMAND	Command to start CATDUAV5 (optional). Default: catstart -run "CATBatchStarter -input @CONFIG@ -output @OUTDIR@"
CMI_ARCHIVE_ALLOW_BROKEN_LINKS	If set to "ON", broken links in CMI Archive are ignored. Default is "OFF" - broken links are not allowed in a CMI Archive.
CMI_PREVENT_DIFFERENT_VERSIONS	If set to "ON", "To Catia" is checked against different versions of same file in Catia and in WB

CMI_DYNAMIC_CATALOG	If set to "ON", CMI stores the Part Master ID in CATCatalogs. Manage Catalogs opens always the latest version of the Part, and not a static revision.
CMI_READ_INERTIA	If set to “ON”, inertias from CATParts and CATProducts will sent to Teamcenter, if set to “ONLY_CATPART”, only the inertias of CATParts will sent to Teamcenter
CMI_CONFINE_INERTIA_TO_MAINBODIES	If set to “ON”, the inertias sent to Teamcenter are confined to those of main part bodies.
CMI_ENABLE_CRE_ANALYSIS_ARCHIVE	If set to "ON", the Button CreateArchive is enabled, if the top-level-node is a CATAnalysis.
CMI_ANALYSIS_ARCHIVE_OWN_WINDOW	If set to "ON", CMIArchives with CATAnalysis as top-level-node will load into an own window in Catia.
CMI_NEW_SYNCHRONIZE	If set to "ON" enable the new SynchronizeCmd and disable the old Synchronize, update and Create functions
CMI_SKIP_RO_TEMPLATE	If set to ON, Template CATProducts that were read as Read-only will be skipped during synchronize, ie. they will not cause a warning
CMI_GETORIGGEO_DESIGNMODE	If set to "ON", GetOrigGeoCmd: If all files of the selected structures are located in the exchange directory ask to load the selected assemblies into DesignMode.
CMI_ENABLE_EDUFLAG_CHECK	If Set to "ON" check for educational flag during Update /Synchronize / Create / SaveAs. If Edu flag is set for a file to be saved, the Save action will be declined
CMI_ENABLE_ACTIVATEDEACTIVATE	If set to "ON" the default value of the user option "Deactivate geometry files not from CMI Workbench" is set to checked. To disable Hide / Show also set CMI_DISABLE_HIDESHOW=ON
CMI_ENABLE_CHECKFOREMBEDDEDCHANGE	If set to "OFF" do not check if there are embedded changes in the structure (Performance) This is permissible if you have not customized the message **`x3UseEmbeddedProduct`** in Teamcenter.
CMI_GEOPOS_NOLINK	Set to "OFF" if you use geometry positions but do not link the document to the part when a CATPart is created (default ON)
CMI_DISABLE_SYNC_PROCESS	If set to "ON" the synchronize button is disabled if the current window contains a CATProcess
CMI_HIDESHOW_ADDTOSESSION	If set to "ON" HideShow / ActivateDeactivate only affects CATIA Windows where the Top-Level Items are available in the current CMI Workbench.
CMI_ENABLE_CREATE_IMPORT_FILE	If set to "ON" the check box "Write Import/Export Mapping File" is available in the Synchronize dialog and a Mapping file can be used in Reconnect Teamcenter
CMI_ENABLE_EXPORTCMD	If set to "ON" the “Export to Folder” Command is available in the CMI Toolbar
CMI_EXPORT_STDATTRIBUTES	To set the CATIA standard attributes Nomenclature, Revision, Description or Definition back to the original value of the initial import set the value of CMI_EXPORT_STDATTRIBUTES to the attributes you want to set back. Only used with the CMI Export Cmd.
CMI_REMOVE_CONTEXTMENU	Set to "ON" to disable the contextual CMI menu in the CATIA Product Structure
CMI_IGNORE_EMBEDDED_LEAFNODE	Set to "ON" to ignore embedded ccomponents with no children
CMI_ENABLE_CMICATALOGINSERTSCRIPTCMD	Set to "ON" to enable the "Insert CATScript from Teamcenter" command

CMI_EXPORT_CUSTOMIZE_NAMING	Set to "ON" to invoke the custom TC methods x3GetExpNamingSchema, x3GetExportRefNames and x3GetExportInstNames. This option must not been set if x3GetExportRefNames and x3GetExportInstNames are not customized.
CMI_ENABLE_CGM	Set to "ON" to enable the Update, Synchronize, Save As and Save for Doc commands for CGM documents in CATIA
CMI_STORE_ANALYSIS	Set to "ON" to store CATAnalysis DataItem in Teamcenter.
CMI_ARCHIVE_BOM_CHILDREN	Set to "ON" to allow Bom structures under a CMIArchive. Default implementations of *x3CreateUsesRel and x0WkBnch:x3GetInstInfos have been changed to discriminate between archive and regular occurrences of an instance.*
CMI_REPLACE_ALLOW_NONBOM	Set to "ON" to allow to replace NonBom geometries with "Replace from Teamcenter". To use this function the NonBom geometries has to be removed from the structure in TC, for instance by setting the CMI server option "CMI_REMOVE_GEOMETRY=DOCUMENT"
CMI_REMOVE_CMIREPLACEPDMSTRUCTURECMD	Set to "ON" to disable "Replace from Teamcenter" functionality
CMI_CALC_BBOX_FOR_IGNOREDCHILDREN	Set to "ON" to calculate the bounding box of configurable ignored children (with configuration BBOX=true) during update of the father product. Set to "FORCE" to update the bounding box whenever the father product is writeable.
CMI_ADDITIONAL_CMIAPPLICATION	By default CMI is only loaded in the CNEXT process. Set CMI_ADDITIONAL_CMIAPPLICATION=<your application> to use CMI in an application other than CNEXT. To get the correct name of the appication use CMI_DEBUG=ON -> "No CMI in Application: <…>"
CMI_ACTIVATEREFERENCEGEOMETRY	Set to "ON" to activate embedded component on Read if a configurable reference geometry is available on this component and the reference is sent by Teamcenter.
CMI_CHECKSAVED2	If set to "ON" use different method to detect modified files. Caution: If used CMI tries much more updates.
CMI_DISABLE_COMPARECMD	Set to "ON" to disable the compare version functionality
CMI_HIDE_COMPARECMD	Set to "ON" to hide corresponding CMI command in toolbar and menu without deactivating the command. It is possible to use the hidden commands via macro.

Hide CMI Commands

You can remove most buttons in the CMI Toolbar with an environment variable (see above). However, if you intend to use a CMI Command in a VB Script, you should not remove the command. Most commands can also be merely hidden with a setting of the form CMI_HIDE_...CMD. CMI commands are not generally intended for script use, and you do so at your own judgement.

Display CATIA Node Name in Synchronize

The width and visibility of columns in the Synchronize dialog can be configured by editing the file `intel_a\resources\msgcatalog\CMIUpdateCreateDialog.CATRsc`

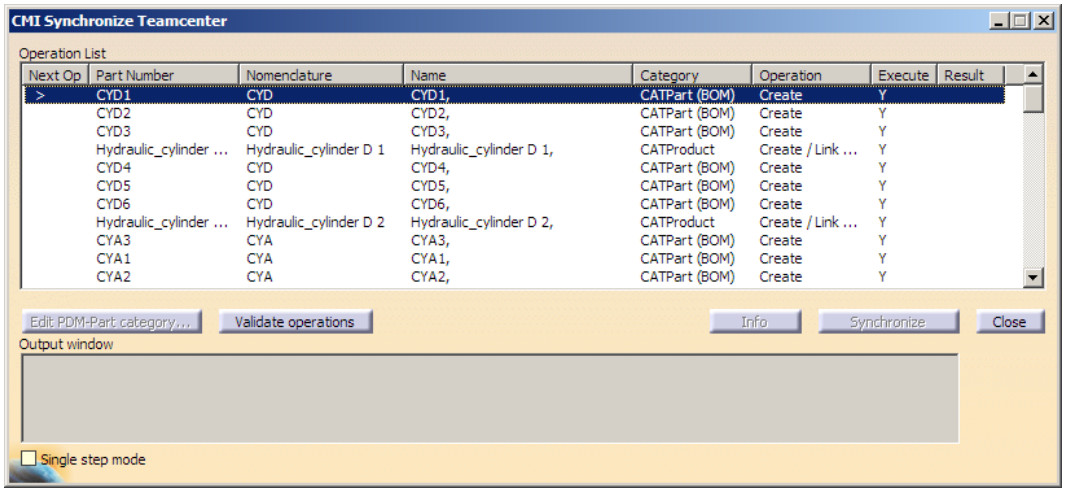
CMI Synchronize Teamcenter
Operation List
Next Op | Part Number | Nomenclature | Name | Category | Operation | Execute | Result
> CYD1 CYD CYD1, CATPart (BOM) Create Y
CYD2 CYD CYD2, CATPart (BOM) Create Y
CYD3 CYD CYD3, CATPart (BOM) Create Y
Hydraulic_cylinder ... Hydraulic_cylinder D 1 Hydraulic_cylinder D 1, CATProduct Create / Link ... Y
CYD4 CYD CYD4, CATPart (BOM) Create Y
CYD5 CYD CYD5, CATPart (BOM) Create Y
CYD6 CYD CYD6, CATPart (BOM) Create Y
Hydraulic_cylinder ... Hydraulic_cylinder D 2 Hydraulic_cylinder D 2, CATProduct Create / Link ... Y
CYA3 CYA CYA3, CATPart (BOM) Create Y
CYA1 CYA CYA1, CATPart (BOM) Create Y
CYA2 CYA CYA2, CATPart (BOM) Create Y
Edit PDM-Part category...
Validate operations
Info
Synchronize
Close
Output window
Single step mode

Figure 32: Synchronize dialog with CATIA node names

E.g. the column ***Name*** is not shown by default, but it contains the text configured in the customized display for `Reference Product`:

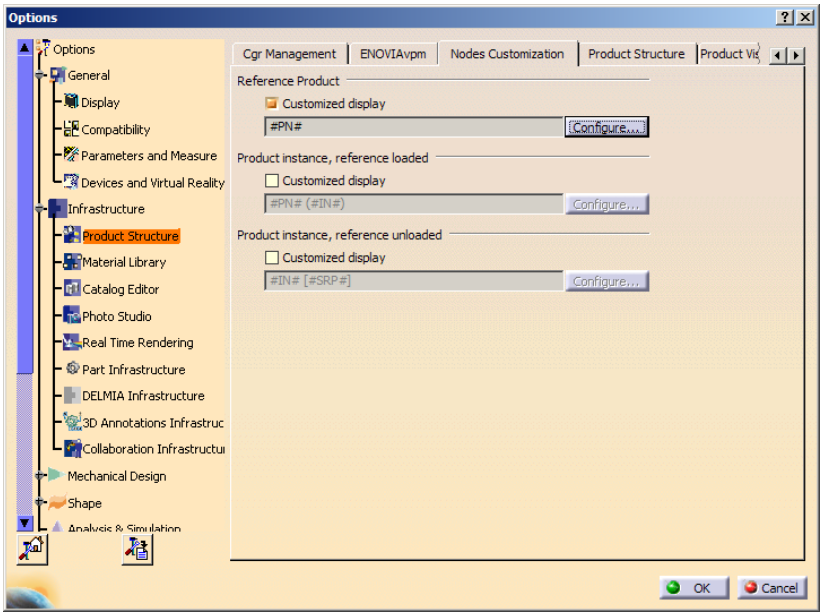

Figure 33: CATIA Options - Nodes Customization

To make the *Name* Column visible by default, the following setting has to be changed in the file `CMIUpdateCreateDialog.CATRsc`:

```
// Column width of Name
JobMultiList.ColumnWidth4 = "0";
```

To hide `Nomenclature` and show `Name` instead you have to set:

```
// Column width of Nomenclature
JobMultiList.ColumnWidth3 = "0";
// Column width of Name
JobMultiList.ColumnWidth4 = "16";
```

CHAPTER 10

Data Models

The following figures show the entire data model of the CATIA Workbench (CMI) and of the structure of the persistent CMI-classes.

Data structure Catia-Workbench

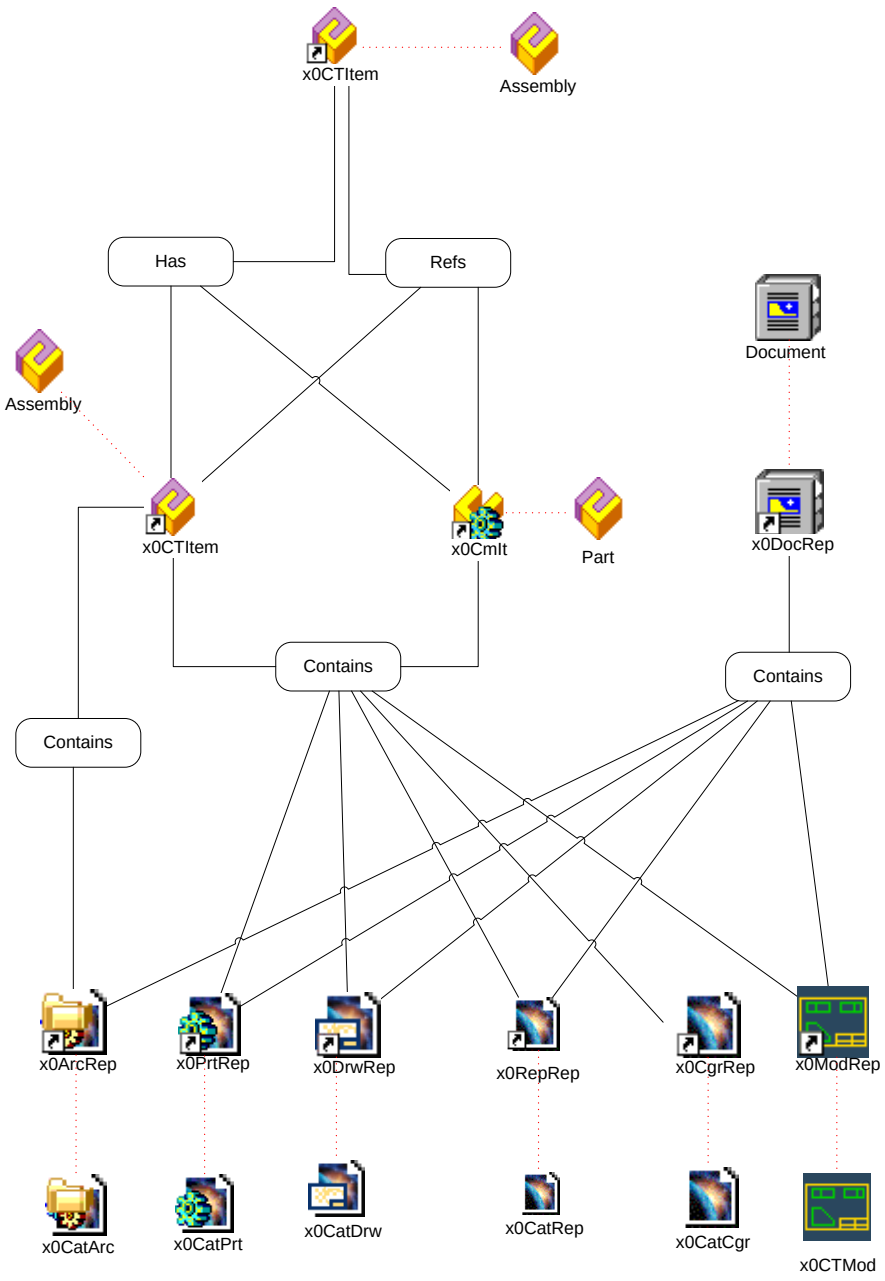

Figure 34: Data Structure of Catia-Workbench

Data structure of CMI-Classes

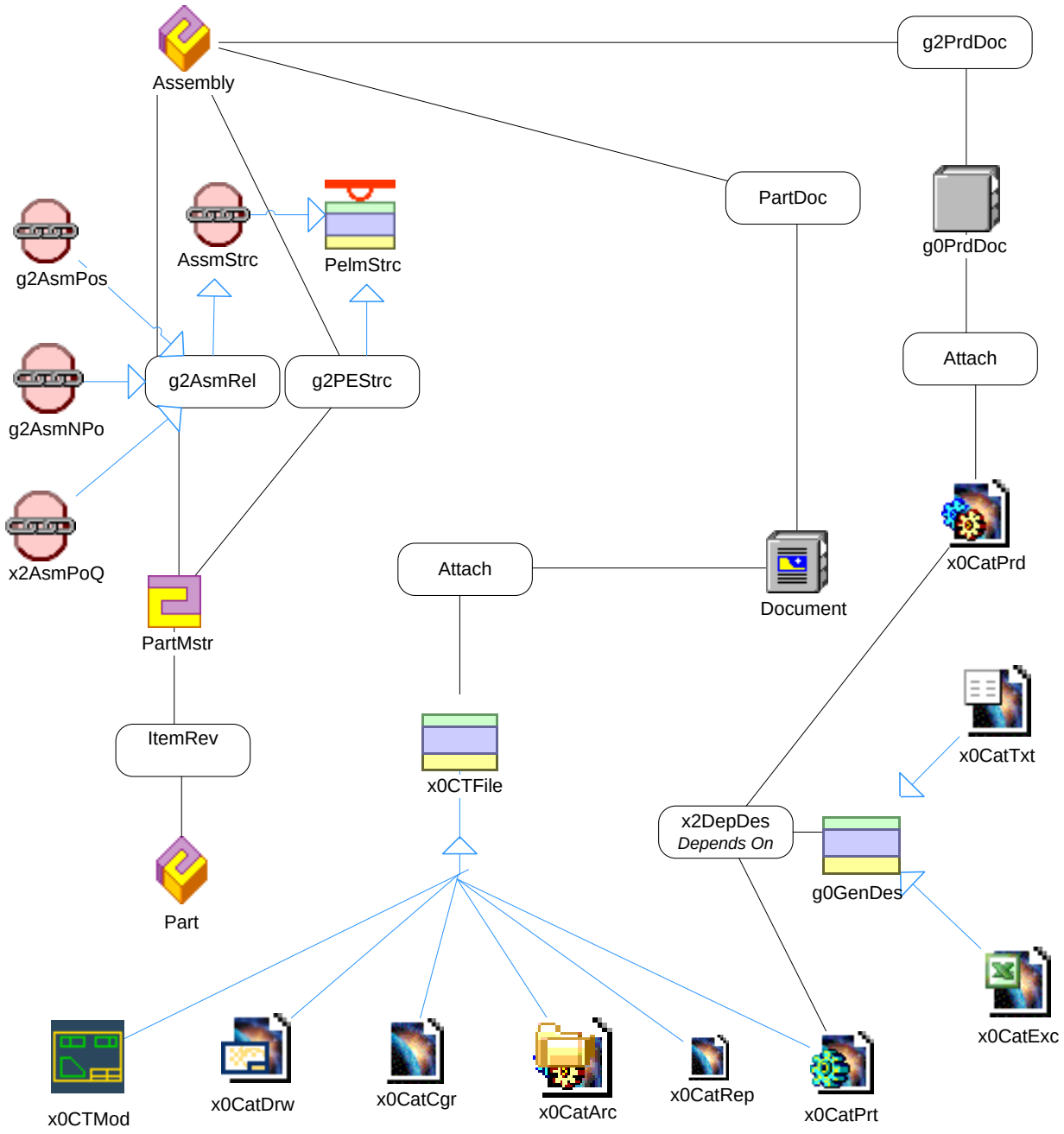

Figure 35: Data Structure of CMI-classes

g0GenBin Class Hierarchy

Figure 36 shows the class hierarchy under the CMI class g0GenBin.

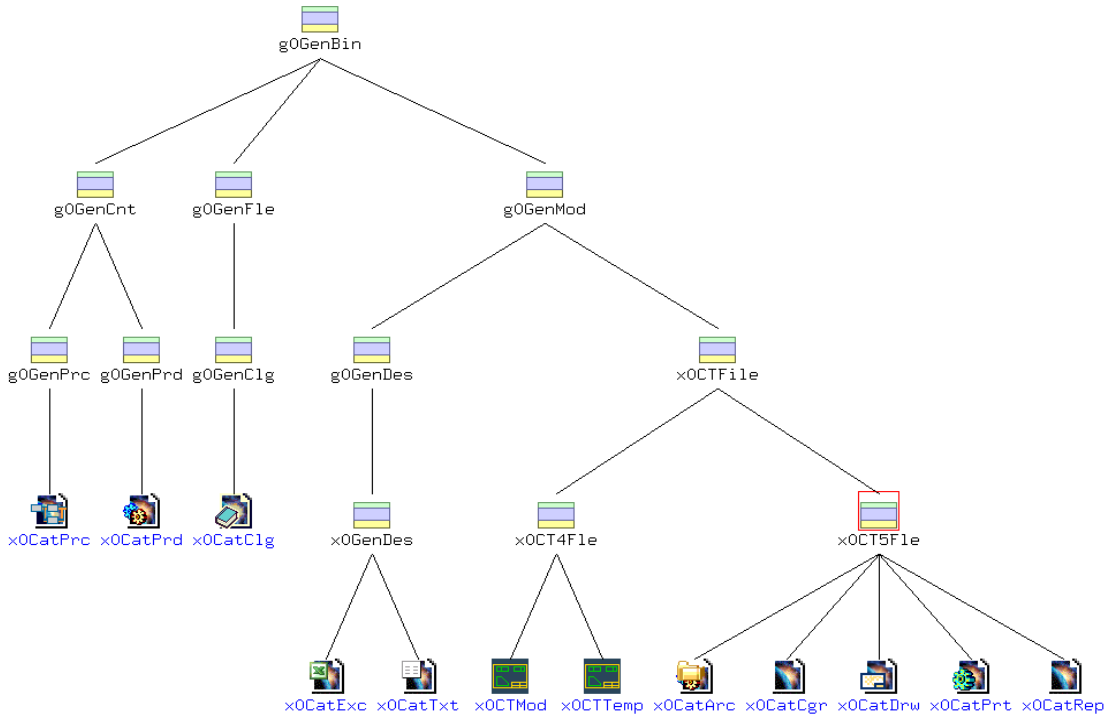

Figure 36: Class hierarchy under g0GenBin

-g0GenBin has the parent class IndepBin and is the container for all CATIA data classes
-g0GenCnt is a child class of g0GenBin and is the container class for the CATIA V5 documents, which are not visible in the CATIA workbench.
-g0GenPrc/x0CatPrc is the data class for the CATIA V5 CATProcess file.
-g0GenPrd/x0CatPrd is the data class for the CATIA V5 CATProduct file.
-g0GenFle/g0GenClg/x0CatClg is the data class for the CATIA V5 Catalog file.
-g0GenMod/g0GenDesx0GenDes/x0CatExc is the data class for a design table in the MS-Excel-Format
-g0GenMod/g0GenDesx0GenDes/ is the data class for a design table in the Textformat.
-g0GenMod/x0CTFile is a child class of g0GenBin and is the container class for the CATIA documents, which can be visible in the CATIA workbench.
-x0Ct4Fle/x0CTMod is the data class for the CATIA V4 model file.
-x0CT4Fle/x0CTTemp is the data class for the CATIA V4 model template file.
-x0CT5Fle/x0CatPrt is the data class for the CATIA V5 CATPart file.
-x0CT5Fle/x0CatDrw is the data class for the CATIA V5 CATDrawing file.
-x0CT5Fle/x0CatCgr is the data class for the CATIA V5 cgr file.
-x0CT5Fle/x0CatArc is the data class for the CMI Archive file.
-x0CT5Fle/x0CatRep is the generic data class for the configurable shape representations.

g0RepItm Class Hierarchy

Figure 36 shows the class hierarchy under the CMI class g0RepItm.

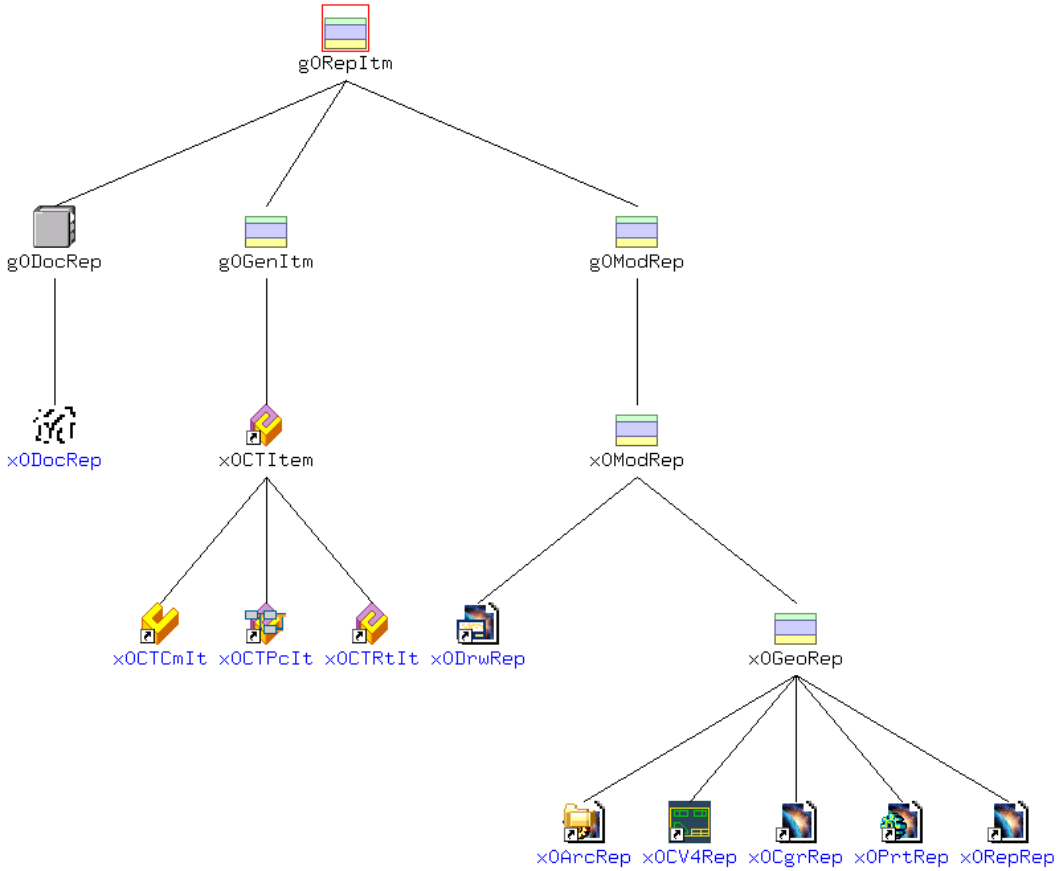

Figure 37: Class hierarchy under g0RepItm

-g0RepItm has the parent class g0PdmItm and is the container for all CATIA representation classes which are visible in the CATIA Workbench.
-g0DocRep/x0DocRep represents a document.
-g0GenItm is the container class for the Part representations.
-x0CTItem is the main Representation class for the Assembly class, x0CTCmIt is the representation class for the Cmponent class. x0CTPcIt is the representation class for a CATProcess root item.
-g0ModRep/x0ModRep is the main container for the Model representations.
-x0DrwRep represents the CATIA V5 Drawing data item class. (CATDrawing).
-x0GeoRep is the container for the geometrical representations.
-x0ArcRep represents the CMIArchive data item class (CMIArchive).
-x0CV4Rep represents the CATIA V4 model data item class (model).
-x0CgrRep represents the CATIA V5 cgr data item class (cgr).
-x0PrtRep represents the CATIA V5 Part data item class (CATPart).
-x0RepRep represents the data item class x0PrtRep.